Universidade do Minho
Instituto de Educação

Jaylson Teixeira

Contribuições para o Ensino de Programação de Computadores a Futuros Professores de Matemática

março de 2017

Contribuições para o Ensino de Programação de Computadores a Futuros Professores de Matemática

Jaylson Teixeira

UMinho | 2017

Universidade do Minho
Instituto de Educação

Jaylson Teixeira

Contribuições para o Ensino de Programação de Computadores a Futuros Professores de Matemática

Tese de Doutoramento em Ciências da Educação
Especialidade em Tecnologia Educativa

Trabalho realizado sob a orientação do
Doutor António José Meneses Osório
e do
Doutor António Luís Valente de Sousa Teixeira

março de 2017

DECLARAÇÃO

Nome **Jaylson Teixeira**

Endereço electrónico: **jaylsont@ufrb.edu.br / jaylsont@gmail.com**

Telefone: **55-75-9-8202-3376** / **55-75-3634-3042**.

Número do Bilhete de Identidade: **RG 12.814.978-4**

Título da tese: **Contribuições para o Ensino de Programação de Computadores a Futuros Professores de Matemática**

Orientador(es): Professor Doutor **António José Meneses Osório**

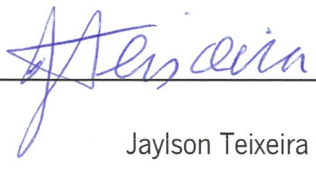
Professor Doutor **António Luís Valente de Sousa Teixeira**

Ano de conclusão: **2017**.

Designação do doutoramento: **Doutoramento em Ciências da Educação**

Especialidade: **Tecnologia Educativa**.

Universidade do Minho, 22/03/2017.



Jaylson Teixeira

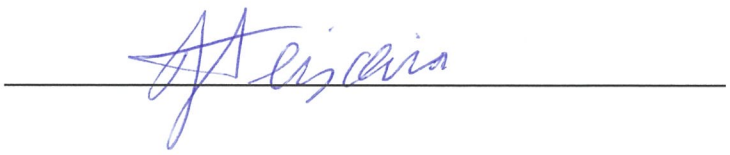
DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração da presente dissertação. Confirmo que em todo o trabalho conducente à sua elaboração não recorri à prática de plágio ou a qualquer forma de falsificação de resultados.

Mais declaro que tomei conhecimento integral do Código de Conduta Ética da Universidade do Minho.

Universidade do Minho, 22 de março de 2017

Jaylson Teixeira



Agradecimentos

Agradeço aos meus orientadores pelas oportunidades oferecidas que me fizeram aprender mais e muito, além da paciência e generosidade com que me receberam.

Agradeço as professoras Rosineide Pereira Mubarack Garcia (UFRB) e Custódia Alexandra Almeida Martins (UMinho) pelo trabalho hercúleo que viabilizou a cooperação entre estas universidades.

Agradeço aos meus alunos e colegas professores de trabalho que muito me ensinaram e me acolheram ao iniciar na UFRB como professor em tempo integral. Em especial meu colega Leandro Diniz que me inspirou a modificar o estilo das minhas provas.

Agradeço a paciência dos meus amigos que, apesar das cobranças, entenderam a minha necessidade de estar recluso neste longo período de tese. Em especial meu querido amigo Paulo Cesar Faria que me apresentou o Scratch.

Agradeço ao meu amigo de longa data, Luís Gonçales Bueno de Camargo, que, mais uma vez, esteve presente quando precisei, desta vez com a revisão dessa tese.

Agradeço ao meu filho Ícaro Mafra Teixeira que me apoiou e sempre me orgulhou, além de ser meu assessor particular para assuntos de tradução.

Agradeço aos meus pais José Antônio Teixeira e Nancy Nepomuceno Teixeira pelo suporte de todas as maneiras e pela insistência e esperança que seu filho ainda tem jeito.

Agradeço minha companheira Luciana Santos Soares pela alegria, espontaneidade, sorriso e bom humor, enfim, por ser ela mesma e estar ao meu lado.

CONTRIBUIÇÕES PARA O ENSINO DE PROGRAMAÇÃO DE COMPUTADORES

A FUTUROS PROFESSORES DE MATEMÁTICA

Jaylson Teixeira

Doutoramento em Ciências da Educação — Especialidade em Tecnologia Educativa

Resumo

Esta investigação se ocupa do estudo do processo de aprendizagem de programação de computadores, com o objetivo de avaliar práticas adotadas para promover o pensamento computacional em um ambiente de aprendizagem para futuros professores de matemática. A metodologia baseou-se num estudo de caso envolvendo alunos de licenciatura em matemática e licenciatura em física na Universidade Federal do Recôncavo da Bahia. O processo estabelecido foi avaliado pelos projetos finais dos alunos, sendo estes os produtos resultantes da aprendizagem. A avaliação foi realizada com os critérios estabelecidos pela ferramenta Dr. Scratch através de uma análise estática dos programas. A taxonomia de Bloom foi utilizada como critério da análise de as atividades de Roteiros, Prova e Projetos.

Entre os resultados mencionados podemos destacar a necessidade de as escolas e universidades incentivarem a autonomia e a criatividade. Os roteiros estabelecem zonas de desenvolvimento proximal diferentes nos alunos, possibilitando que um aluno mais apto ajude um menos apto. À medida que se sucedem os roteiros, os alunos vão melhorando competências como proficiência em programação e autonomia. As provas devem ser adaptadas para mídia lápis e papel, solicitando o entendimento de programas por escrito, possibilitando uma avaliação qualitativamente melhor do ponto de vista cognitivo se comparado com a resolução de exercícios. Os projetos são o ponto alto da avaliação. É através dos projetos que notamos a proficiência em programação, a autonomia e a criatividade de forma articulada. Resulta também em metacognição e colaboração, uma vez que os alunos se comparam a seus colegas ao ajudar e serem ajudados. Como conclusão tem-se que o processo de aprendizagem estabelecido contribui para o pensamento computacional quando avaliado pela ferramenta Dr. Scratch e acessa os níveis de cognição mais elevados segundo a taxonomia de Bloom. Ao atingir tais níveis na taxonomia de Bloom, também se pratica o pensamento computacional. Este trabalho deve contribuir para professores e projetistas de currículo interessados no ensino de programação, em especial para aqueles que trabalham nas escolas e em cursos de formação de professores de matemática e ciências para promover projetos e estabelecer planos de ensino que estimulem competências relacionadas ao pensamento computacional, em especial a autonomia e criatividade.

Palavras-chave: Pensamento Computacional; Ensino de Matemática; Formação de Professores.

CONTRIBUTIONS TO THE COMPUTER PROGRAMMING TEACHING FOR FUTURE MATHEMATICS TEACHERS

Jaylson Teixeira

PhD in Education - Specialization in Educational Technology

Abstract

This research deals with the study of the process of learning computer programming, in order to assess practices to promote computational thinking in a learning environment for future mathamatics teachers. The methodology was based on case-stady involving undergraduate students in teaching mathematics and in teaching physics at the Federal University of Bahia Reconcavo. The established process was evaluated by the final projects of students, which are the products of learning. The evaluation was performed with the criteria established by Dr. Scratch tool through a static analysis of programs. Bloom's taxonomy was used as a criteria on Proceedings, Exam and Projects analysis.

Among the mentioned results we can highlight the need for schools and universities to encourage autonomy and creativity. The proceedings establish different zones of proximal development among the students enabling a more effective person helps less one. As the proceedings follow, students will improve skills such as proficiency in programming and autonomy. The exam must be adapted to pencil and paper media, requesting understanding of writing programs, enabling a qualitatively better assessment of the cognitive point of view compared with the resolution of exercises. The projects are the high point of the evaluation. It is through the projects we noticed proficiency in programming, autonomy and creativity in an articulated manner. It also results in metacognition and collaboration, as students compare to their peers while help and be helped. In conclusion, the learning process established contributes to computational thinking when evaluated by Dr. Scratch tool and access the higher cognitive levels according to Bloom's taxonomy. By achieving such levels in Bloom's taxonomy, students also practiced computational thinking. This work should help teachers and curriculum designers interested in educational programming, especially for those working in schools and math and science teacher training courses to promote projects and establish educational plans that encourage skills related to computational thinking, particularly autonomy and creativity.

Keywords: Computational Thinking; Mathematics Teaching; Teacher training.

Índice

Índice de Figuras

Índice de Tabelas

1 Introdução

Este trabalho de investigação está relacionado com as aprendizagens possibilitadas pelas práticas adotadas em um curso de programação de computadores. Em uma sociedade onde o computador está cada vez mais presente e disponível, cabe a nós utilizá-lo de forma criativa e como ferramenta de resolução dos problemas que se apresentam à sociedade. Sendo assim, não podemos aceitar um ambiente de aprendizado meramente instrucional sem dar voz aos alunos. Estes alunos, por sua vez, devem ser capazes de utilizar a tecnologia de forma autônoma e criativa, aspectos que também devem ser apreendidos e valorizados.

O resultado desta investigação deverá interessar aos professores que ensinam programação de computadores para iniciantes, tanto nas escolas quanto na academia, assim como a pesquisadores que se interessam pela aprendizagem de programação e do pensamento computacional. Quero, previamente, esclarecer que o curso de licenciatura em matemática no Brasil se destina a formar professores que atuarão no ensino fundamental e médio. Ensino fundamental é a primeira etapa da educação básica, tem duração de 9 anos, sendo a matrícula obrigatória para todas as crianças com idade entre 6 e 14 anos. Ensino médio corresponde à segunda e última fase da educação básica, tem duração de 3 anos. O ensino médio tem por finalidade aprofundar os conhecimentos adquiridos no ensino fundamental, além de formar o cidadão para a vida social e para o mercado de trabalho. Oferece o conhecimento básico necessário para o estudante ingressar no ensino superior.

No Brasil, ensino básico se refere ao conjunto do ensino fundamental mais o ensino médio. Em Portugal, o termo ensino básico se refere ao que no Brasil seria equivalente ao ensino fundamental. O termo ensino básico no Brasil equivale ao que se chama ensino obrigatório em Portugal. Para melhor compreensão do texto tanto por brasileiros como portugueses, evitarei o uso do temo ensino básico. Veja Tabela 1.

Tabela 1 – Níveis de ensino no Brasil e em Portugal

Idade	Designação no Brasil		Designação em Portugal		Duração
6 –14	Ensino Básico	Ensino Fundamental	(Ensino Obrigatório)	Ensino Básico	9 anos
15–17		Ensino Médio		Ensino Secundário	3 anos

No Brasil existem dois tipos de cursos de graduação em matemática denominados bacharelado e licenciatura, ambos com duração de 4 ou 5 anos. O curso de bacharelado se destina a formar investigadores em matemática para a comunidade científica. O curso de licenciatura em matemática se destina à formação de professores de matemática para ensinar nos níveis fundamental e médio.

Este trabalho constitui-se de seis capítulos, sendo este primeiro a Introdução, no qual exponho o tema da investigação, os objetivos, a justificativa e a motivação para este trabalho. No segundo capítulo abordo o pensamento computacional, o seu reconhecimento como competência fundamental para o século XXI e como a programação e a matemática podem contribuir para ele. No terceiro capítulo apresento uma revisão da literatura que utilizarei para me basear e analisar o processo de aprendizagem de programação. No quarto capítulo apresento o contexto, os envolvidos e os detalhes da metodologia que foi utilizada na investigação. No quinto capítulo apresento os dados recolhidos e procedo à sua análise. Por fim, no sexto capítulo, faço as considerações finais com relação à investigação realizada.

1.1 Contexto e motivação

Obtive a licenciatura em matemática no ano de 1991 e segui uma carreira profissional desenvolvendo *software* para a iniciativa privada com diversas tecnologias, chegando a fazer mestrado em Informática, com especialização em engenharia de software. Engenharia de software se ocupa de várias fases e aspectos de um projeto de software. Em 2010 iniciei uma nova perspectiva profissional, sendo responsável por ministrar a disciplina de Introdução à Lógica de Programação para os cursos de licenciatura em matemática e de licenciatura em física, no Centro de Formação de Professores (CFP) na Universidade Federal do Recôncavo da Bahia (UFRB). Naquele instante constatei que pouco se modificou no ensino de programação para os cursos de licenciatura desde a minha graduação. Nas aulas utilizava-se uma linguagem hipotética conhecida como Portugol que parecia a linguagem de programação PASCAL traduzida para o português. Os conceitos eram apresentados em aulas expositivas e os exercícios eram realizados com tecnologia de papel e lápis. Do meio para o fim da disciplina era introduzida uma linguagem real de programação (linguagem C ou PASCAL) gerando programas em consola de caracteres com tela preta semelhante ao MS-DOS. Os exercícios propostos eram essencialmente numéricos com enunciados do tipo “Apresente os números primos entre 1 e 100”, ”Gere a sequência de Fibonacci”, “Descubra o valor máximo em vetor dado”.

Minha experiência profissional, tanto na iniciativa privada quanto no meio acadêmico, me levou a acreditar que a forma tradicional do ensino de programação de computadores, baseada na metodologia e nas tecnologias dos anos 1980 nos cursos superiores no Brasil, fosse a regra. Há razões para supor que a disciplina é repetida pelos professores tal qual a aprenderam com seus mestres. Esta situação me causou diversas inquietações. Não poderíamos usar interfaces com multimídias, incluindo sons, imagens e animações?

Trabalhando como desenvolvedor de *software*, observei que o conceito de orientação a objetos é largamente utilizado desde a década de 1990. Este conceito não deveria ser apresentado aos alunos? Além disso, os alunos se mostravam bastante irritados com os erros de sintaxe que apareciam em grande quantidade, os quais deveriam ser superados antes de se ocuparem com os algoritmos. Daí em diante venho refletindo e modificando a minha forma de ensinar essa disciplina. Comecei utilizando os computadores desde a primeira aula, acreditando que a prática da programação, possibilitando experimentação, melhora o entendimento. Depois passei a utilizar o Scratch, evitando os erros de sintaxe e podendo direcionar melhor a atenção dos alunos para os algoritmos. O Scratch também possibilitou o uso de recursos multimídia, a orientação a objetos e a execução de algoritmos em paralelo, com relativa facilidade.

A partir desse ponto, outros questionamentos passaram a existir: por que ensino a programar no computador e avalio em uma prova na mídia papel e lápis? Como posso modificar esta relação? Como os programas são feitos em equipe, qual o melhor balanço entre a avaliação coletiva e a individual? Os alunos modificam os cenários e personagens mostrando criatividade e senso estético, usando esta mídia como forma de expressão e diversão, como isso influencia no processo de aprendizagem?

Embora esta pesquisa seja feita em turmas de uma universidade em uma cidade e contexto particulares, ela parte de um modelo de aula tradicional largamente utilizado. Sendo assim, acredito que os resultados locais sejam úteis em várias situações semelhantes existentes no Brasil e noutros países.

1.2 Problema, objetivos e justificativa

O aprendizado de programação de computadores desenvolve o pensamento. Ao programar, o aluno precisa utilizar formas elevadas de pensamento, começando por conceber uma proposta do que será o programa. Entre a proposta e a programação final, ele deve

conceber um modelo que seja implementável em um computador e codificar este modelo em uma linguagem de programação, com o rigor matemático dessa linguagem. Ao perceber que seu programa não está funcionando a contento, ele volta à linguagem, depurando o programa e, muitas vezes, depurando o modelo que está representado na linguagem. Em última instância, o aluno depura seus próprios pensamentos. Desse modo, a programação de computadores é uma oportunidade para promover a aprendizagem do pensamento computacional, desenvolvendo também outras capacidades relevantes do ponto de vista social.

O professor de matemática, visando à formação do aluno para a sociedade, deve promover o aprendizado de capacidades de comunicação, de resolução de problemas, de tomada de decisões, de inferir, de criar, de aperfeiçoar conhecimentos e valores, de trabalhar cooperativamente (Ministério da Educação, 2000, 2002). Estas mesmas capacidades são requeridas pelo pensamento computacional. Quando os Parâmetros Curriculares Nacionais (PCN) para o Ensino Médio do Brasil entram em mais detalhes, especificando as competências e habilidades a serem desenvolvidas em matemática, apresentam estas competências e habilidades agrupadas em 3 classes, chamadas de Capacidades Gerais: representação e comunicação; contextualização sócio-cultural; investigação e compreensão (Ministério da Educação, 2000, 2002).

A representação e a comunicação devem ser desenvolvidas, possibilitando ler, interpretar e entender símbolos matemáticos, em suas diversas linguagens simbólicas, como equações, gráficos, diagramas, fórmulas, tabelas, entre outros. A linguagem de programação pode ser considerada mais um sistema de símbolos que permite a expressão do indivíduo. Os algoritmos permitem uma descrição passo a passo que pode servir de degrau intermediário entre a linguagem corrente e a linguagem matemática (Barcelos & Silveira, 2012). A linguagem de programação pode ser interpretada pelo computador que é uma máquina de inferências matemáticas. Sendo assim, o retorno (*feedback*) que o computador dá ao executar a linguagem de programação ajuda o aluno a entender o mecanismo de inferência. Através da programação é possível criar modelos para interpretar e intervir no real, o que caracteriza uma capacidade desejada ao se referir às capacidades e habilidades de contextualização socioculturais. Já as capacidades e habilidades classificadas como investigação e compreensão, todas as mencionadas nos PCN para o ensino de matemática, podem ser atribuídas ao pensamento computacional. São elas:

> Identificar o problema (compreender enunciados, formular questões etc.); Procurar, selecionar e interpretar informações relativas ao problema; Formular hipóteses e prever resultados; Selecionar estratégias de resolução de problemas; Interpretar e criticar resultados numa situação concreta; Distinguir e utilizar raciocínios dedutivos e indutivos; Fazer e validar conjecturas, experimentando, recorrendo a modelos, esboços, fatos conhecidos, relações e propriedades; Discutir ideias e produzir argumentos convincentes. (Ministério da Educação, 2000, p. 46)

Os PCN estão de acordo com o relatório da comissão de educação para o século XXI da UNESCO, que propõe quatro pilares para a educação: Aprender a Conhecer, Aprender a Fazer, Aprender a viver com os outros e Aprender a ser (International Commission on Education for the Twenty-first Century, 1996).

Do ponto de vista teórico, é razoável conceber que o ensino de programação vise ao pensamento computacional, estando de acordo com os PCN de matemática. Porém, não é por força de uma teoria ou lei que a realidade se estabelece nas escolas. Fatores políticos, sociais e econômicos, assim como as concepções hegemônicas sobre a educação, podem determinar mais a realidade do que boas leis, recomendações ou teorias. Os PCN admitem esta disparidade ao afirmar que:

> Na elaboração de propostas educacionais, além de se considerarem as variáveis regionais, de sentido cultural e socioeconômico, tão significativas em um país de dimensões e de contrastes sociais como o Brasil, é preciso ter clareza de que as propostas, oficiais ou não, na melhor das hipóteses são o início de um processo de transformação, de reacomodação e de readequação. (Ministério da Educação, 2000, p. 47)

O professor, ao se dispor a mudar o ambiente de aprendizado, seja por acreditar na teoria ou pelo dever de seguir as recomendações institucionais, terá que começar experimentando práticas, sem muita confiança e sem uma referência pragmática, uma vez que existem muitos trabalhos orientando sobre as capacidades que se devem promover, mas poucos mostrando como fazê-lo. Este trabalho propõe uma alternativa de como fazer para promover um ambiente de aprendizado através de instrumentos e procedimentos que levam a um melhor ambiente de aprendizado em turmas regulares de licenciatura em matemática. Deste modo enuncio a seguinte questão de investigação:

Como conduzir um processo de aprendizagem em salas regulares de programação de computadores, de modo a contribuir para o desenvolvimento do pensamento computacional na formação de futuros professores de matemática?

Procuro um ambiente de aprendizagem que possibilite o desenvolvimento do pensamento computacional, influenciando o comportamento dos alunos, tornando-os mais engajados no seu próprio aprendizado de forma ativa e criativa. Espero com isso que as aulas se tornem mais úteis e interessantes para os alunos do curso de licenciatura em matemática. Assim, anuncio como objetivo geral: Avaliar as influências das sucessivas práticas adotadas no desenvolvimento do pensamento computacional em um ambiente de aprendizagem para futuros professores de matemática.

O processo de aprendizagem se inicia ao utilizar os instrumentos e procedimentos caracterizando as práticas adotadas. Estas práticas e suas intenções de ensino, relativas a um conteúdo específico ou a valores e comportamentos, criam um melhor ambiente de aprendizado. Pretendo usar as concepções de ensino de matemática e os conteúdos da matemática do ensino médio. Também utilizarei concepções da área de ciência da computação, como a programação orientada a objetos e a valorização do pensamento computacional.

Para me assegurar de estar na posse dos conhecimentos mais relevantes, considerando os meus objetivos de investigação e de desenvolvimento do trabalho, pesquisei trabalhos recentes, com menos de 5 anos, que tratavam de assuntos próximos ao meu interesse. De entre os trabalhos científicos que se referem ao uso de Scratch no ensino fundamental elegi os que melhor se enquadravam no meu contexto (T. F. M. Correia, 2013; T. M. B. F. Dias, 2014; Gomes, 2013; Gregg, 2014; Martins, 2012; M. L. S. Oliveira, Souza, Barbosa, & Barreiros, 2014; A. S. Pinto, 2010; B. F. R. Santos, 2014; Sousa & Lencastre, 2013, 2014) e outros que se referem ao ensino médio (Aldana-Avilés, 2015; Alencar, Freitas, & Danielle, 2014; Aureliano & Tedesco, 2012; Marques, 2013; R. M. S. Santos, 2013; Scaico et al., 2013), porém, encontrei apenas um trabalho recente sobre Scratch com alunos de licenciatura em matemática (Mendonça, 2010). Contudo, este trabalho não se refere ao ensino regular da disciplina de programação e sim a um curso de extensão universitária. Encontrei apenas um outro trabalho no ensino regular de programação superior, para alunos de licenciatura em computação (Dias & Serrão, 2014). Deste modo o meu trabalho também se justifica por preencher uma lacuna na literatura no que diz respeito ao uso do Scratch no ensino superior para formação de professores que atuarão nas escolas em um futuro próximo. Espero que este trabalho inspire os professores e revisores do currículo a adotar, na disciplina de programação, o paradigma da programação orientada a objetos, interface gráfica e linguagem Scratch, encontrando neste trabalho um suporte para iniciar e levar adiante estas ideias.

Do ponto de vista social e político, este trabalho se justifica na medida em que estimula o pensamento computacional, defendendo o desenvolvimento de capacidades necessárias para a sociedade dos nossos dias, levando a ganhos econômicos e sociais para o país, conforme menciona o relatório da Comissão Internacional sobre Educação para o Século XXI da UNESCO (International Commission on Education for the Twenty-first Century, 1996) e os Parâmetros Curriculares Nacionais do Brasil (Ministério da Educação, 2000, 2002).

Do ponto de vista pessoal, eu não estava satisfeito com os resultados da aprendizagem de programação que meus alunos evidenciavam, principalmente pelo fato de não serem capazes de criar um programa, da concepção aos testes finais, com relativa autonomia. Diante de tantas possibilidades abertas pela programação de computadores, me incomodava que eles não deixassem a imaginação fluir com entusiasmo. Menciono a relativa autonomia, uma vez que o aluno deve contar com o professor e seus colegas para superar dificuldades, fazendo parte da sua autonomia, a metacognição que o faz reconhecer suas limitações, isolar o problema e procurar ajuda quando necessário. Espero ainda que, como futuros professores, eles criem uma consciência da importância da autonomia e da criatividade, incorporando estes valores em suas aulas, quando estiverem atuando nas escolas.

1.3 Organização do trabalho de investigação

Este estudo se deu com 10 turmas regulares de Introdução à Lógica de Programação durante os anos de 2011 a 2014, com 226 alunos, totalizando 88 projetos finais e 600 horas-aulas de observação. Utilizei uma abordagem qualitativa associada a um paradigma sócio crítico. A investigação é qualitativa, uma vez que ela é construída de forma indutiva, a partir dos dados obtidos no ambiente natural, levando em conta seus significados, e é sócio-crítica na medida em que existe um componente ideológico com a intenção de que os alunos se tornem pessoas mais autônomas e criativas. A metodologia utilizada enquadra-se no estudo de caso, que refletindo o método indutivo sobre o qual a pesquisa foi historicamente construída, em ciclos de observação, planejamento, intervenção e avaliação. Para confirmar os resultados da autobiografia (triangulação), foi realizado no estudo de caso considerando as diversas interações da dos anos de 1011 a 2014 como instâncias de um mesmo caso, na intenção de analisar o processo de aprendizagem.

A coleta de dados foi feita através do estudo autobiográfico e através de análise documental para a triangulação no estudo de caso. Dos 88 projetos desenvolvidos utilizei 10, selecionados segundo o desempenho na disciplina medido por linhas de código (LOC - Lines Of Code) e pela complexidade ciclomática de Mac Cabe. Com base na distribuição de frequência seguindo o indicador de linhas de código e complexidade de Mac Cabe, selecioneis 5 projetos que estavam abaixo das médias desses indices e mais 5 projetos acima dessas médias. Estes projetos serviram para avaliação do pensamento computacional, de acordo com uma análise estática do código, baseada na ferramenta Dr. Scratch (http://drscratch.programamos.es), desenvolvida pela Universidad Rey Juan Carlos. Os documentos que representam o processo utilizado na disciplina, por sua vez, serão analisados à luz da taxonomia cognitiva de Bloom. Para posicionar minha investigação relativamente aos demais trabalhos científicos recentes, busquei teses de doutoramento dos últimos 5 anos em bases de dados ou agregadores através dos *sites* Open Access Infrastructure for Research in Europe (OpenAIRE), Repositório Científico de Acesso Aberto de Portugal (RCAAP), RECOLECTA (Recolector de Ciencia Abierta), Networked Digital Library of Theses and Dissertations (NDLTD), Banco de Teses da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes). Foram encontradas três teses nas áreas de educação, educação matemática e ciências da computação com temas semelhantes. Cynthia Collins Selby (2014) realizou seus estudos na área de educação *How Can the Teaching of Programming Be Used to Enhance Computational Thinking Skills?.* Selby faz um estudo baseado na Teoria Fundamentada em Dados (Grounded Theory) a partir de uma comunidade *online* com interesses em pensamento computacional, resolução de problemas e programação de computadores. A partir das concepções de 143 indivíduos, relatadas em questões no fórum de discussão, 42 questionários respondidos e 10 entrevistas, emergiu do estudo uma taxonomia do pensamento computacional representada pelas atividades de generalização, decomposição, abstração funcional, abstração de dados, projeto de algoritmos e avaliação, sendo esta a ordem crescente de dificuldades percebida. Este trabalho se assemelha ao meu ao tratar do ensino de programação visando o pensamento computacional e difere do meu trabalho principalmente por ser de ordem teórica, apresentando uma taxonomia, mas não indicando como utilizá-la, enquanto eu procuro focar a prática do professor em sala de aula para estimular o pensamento computacional a partir da prática da programação. Rodrigo Dalla Vecchia (2012) desenvolveu um estudo na área de educação matemática com o título *A Modelagem Matemática e a Realidade do Mundo Cibernético*. O trabalho de Dalla Vecchia se assemelha ao meu na medida

em que usa o Scratch para construção de jogos eletrônicos, porém ele adota a perspectiva da modelagem para o ensino de matemática. Para Vecchia a linguagem Scratch descreve o modelo que deve ser criado e aperfeiçoado para resolver problemas propostos pelos alunos. A experiência foi realizada em um curso de extensão para alunos de licenciatura em matemática em 8 encontros de 4 horas, totalizando 24 horas de aula para 14 alunos dos quais 4 duplas foram observadas detalhadamente. Andréa Pereira Mendonça (2010) realizou a sua investigação na área de ciência da computação designada *Programação Orientada ao Problema: Uma Metodologia para Entendimento de Problemas e Especificação no Contexto de Ensino de Programação para Iniciantes* , focando-se nos alunos iniciantes de ciência da computação. Este trabalho assemelha-se ao meu na medida em que o processo pode ser reproduzido com os documentos disponíveis no apêndice e também se vale de classes regulares de ensino de programação, visando a ensinar não só técnicas de programação, mas também valores. No caso de Mendonça, valores associados à engenharia de software e no meu caso valores relacionados ao pensamento computacional. Na pesquisa de Mendonça são analisadas duas turmas num total de 70 alunos, que estimo em 60 horas de aula. Mendonça vê o ensino de programação como um caso de engenharia de *software*, devendo o programador ter uma visão mais abrangente, contextualizando o programa solicitado e lidando com requisitos incompletos e ambíguos. Os alunos também devem se preocupar com os testes a serem executados sobre o seu produto. Eles devem questionar para desenvolver um produto de *software* de referência cujos requisitos devem ser elicitados (elicitados = levantados + descobertos). Os produtos têm implementações diferentes, mas se assemelham, pois são criados a partir de uma referência. Não é solicitado documento de planejamento (*design*). No trabalho que eu realizei, os requisitos são determinados pelos alunos, muitas vezes durante o processo, dando margem à criatividade e gerando produtos de *software* distintos. Embora eu não solicite um documento de requisitos, solicito um documento de planejamento – o *storyboard*.

Ao comparar a minha investigação com as outras três semelhantes e recentes, observei que o número de turmas, quantidade de alunos envolvidos e horas-aulas se mostraram bem mais elevados no meu trabalho, o que contribuiu para minhas observações de campo serem mais alargadas, tal como se pode verificar através da Tabela 2

Tabela 2 - Teses Recentes e Parâmetros Geradores de Dados

Título	Área	N.º de Turmas	Número de Envolvidos	Horas-Aula de Observação	Autor
How Can the Teaching of Programming Be Used to Enhance Computational Thinking Skills?	Educação	Uma comunidade online	143	Não se Aplica	Selby (2014)
A Modelagem Matemática e a Realidade do Mundo Cibernético	Educação Matemática	1	14	34	Dalla Vecchia (2012)
Programação Orientada ao Problema: Uma Metodologia para Entendimento de Problemas e Especificação no Contexto de Ensino de Programação para Iniciantes	Ciência da Computação	2	70	60	Mendonça (2010)
Contribuições do Ensino de Programação de Computadores a Futuros Professores de Matemática	Tecnologia Educativa	10	226	600	Jaylson Teixeira (2016)

2 Tecnologias e pensamento: uma dialética poderosa

Este capítulo tem como finalidade compreender a noção de pensamento computacional, destacar a sua importância para os alunos na fase escolar e para a necessidade de formação de professores de matemática com competência para estimular esta forma de pensar.

2.1 O que é Pensamento Computacional?

O computador está cada vez mais presente nas nossas atividades sociais. Apenas operar o computador já não é mais suficiente para as competências exigidas pela sociedade atual. A sociedade demanda o conhecimento de computação como disciplina básica, ao lado da matemática e das ciências (Tucker et al., 2003). Aprender computação vai além de usar o computador, assim como saber astronomia é mais que usar um telescópio (Denning, 2010). Por isso o conceito de pensamento computacional é importante para a nossa geração e cada vez mais importante para as gerações futuras.

O termo pensamento computacional foi amplamente divulgado desde 2006 a partir do trabalho de Jeannette Wing intitulado "Computational thinking". Com esse artigo o termo que Seymour Papert tinha já utilizado em 1980 em *Mindstorms*, ganhou força e provocou discussão na comunidade científica. Wing propõe o pensamento computacional como uma abordagem para resolver problemas, conceber sistemas e compreender o comportamento humano que se baseia em conceitos fundamentais na ciência da computação. Tem como estratégia o uso de abstrações e decomposição de problemas complexos em partes mais simples. É uma maneira como as pessoas pensam de modo a resolver problemas com a parceria de um computador (Wing, 2006). Nos anos que se seguiram, Wing reformulou a definição à medida que pensamento computacional foi ganhando maturidade na comunidade científica. Em 2008, Wing expõe uma definição que destaca as origens da computação a partir da matemática, da engenharia e das ciências:

> Pensamento computacional é um tipo de pensamento analítico. Ele compartilha com o pensamento matemático de forma geral na medida em que aborda a resolução de problemas. Ele compartilha com o pensamento de engenharia [isto é, a maneira de pensar dos engenheiros para resolver problemas] de forma geral na medida em que aborda concepção e avaliação de um sistema grande e complexo que opera dentro das limitações do mundo real. Ele compartilha com o pensamento científico de forma geral na medida em que aborda o entendimento da computabilidade, compreensão, inteligência e o comportamento da mente e do humano. (Wing, 2008, p. 1)

Em 2014, Wing consegue sintetizar a definição de pensamento computacional destacando que o processo computacional não precisa ser necessariamente realizado por um computador:

> O pensamento computacional são os processos de pensamento que envolvem a formulação de um problema expressando suas soluções, de tal forma que um computador ou humano possam executar de forma efetiva. (Wing, 2014, p. 1)

Eduard Fox, professor de ciência da computação no Virginia Polytechnic Institute and State, citado por Lin (National Research Council, 2010) sugere uma origem remota do pensamento computacional ao afirmar que ele já estava presente na espécie humana através das histórias orais que nossos antepassados contavam e com as quais modelaram e representaram a realidade e passaram esse saber para as gerações seguintes, enriquecendo esses modelos para levar adiante a exploração, a descoberta e a sustentação da vida (National Research Council, 2010).

Peter Denning explica que o termo pensamento computacional evoluiu do que se chamava pensamento algorítmico nos anos 1950 e 1960, que significava uma orientação mental para formular um problema como uma conversão de entrada em saída, procurando um algoritmo que desempenhasse esta conversão. Hoje, este termo foi expandido para agregar o pensamento com abstrações hierarquizadas, isto é, abstrações que são definidas e definem outros níveis de abstração. Também incluiu o uso da matemática para desenvolver os algoritmos e examinar a solução para diferentes escalas considerando o tamanho dos problemas (Denning, 2009).

Denning conta que este movimento promovido pelos cientistas da computação emerge de um esforço para esclarecer que ciência da computação é mais do que o desenvolvimento de programas e computadores. No entanto, o pensamento computacional já era utilizado nas demais ciências muito antes dos cientistas da computação terem se apropriado do conceito. Em 1940, John von Neumann dizia que o computador seria não apenas uma ferramenta para a ciência mas seria uma maneira de fazer ciência. Em 1975 o prêmio Nobel de física Ken Wilson promoveu a ideia de que simulação e computação eram maneiras de se fazer ciência que não estavam facilmente disponíveis antes da invenção do computador. Wilson argumentava que computação se tornara o terceiro pilar da ciência, junto com a teoria e a experimentação. O termo pensamento computacional era comum em seus discursos (Denning, 2009).

Oleg K. Tikhomirov, usando o conceito de mediação da teoria sócio-histórica comenta, já em 1981, que o computador é uma nova forma de mediação do homem com seu meio. Esta nova ferramenta traz uma nova maneira de pensar, na medida em que os processos externos

são apropriados pelo homem e passam a fazer parte da estrutura interna dos seus pensamentos (Tikhomirov, 1981).

Seymour Papert, em seu livro *Mindstorms*, mostra como a programação, utilizando a linguagem LOGO, pode ser útil para se aprender matemática. Conceitos como variáveis e recursividade são apresentados de forma natural. Princípios como depuração (*debugging*), construção de estruturas complexas a partir da reutilização de partes menores e mais simples, ou decomposição de um todo complexo em partes mais simples, são encontrados nos desafios que os alunos superam ao programar em LOGO. Para interligar estas estruturas complexas, é necessário criar abstrações em diversos níveis. Papert cita esta atividade de programação como um exemplo tangível de resolução de problemas, tal como sugere George Pólya em seu livro *A Arte de Resolver Problemas*. Papert afirma que, ao invés de cair no maniqueísmo da educação tradicional, quando se tem uma resposta certa ou errada na programação, o erro (*bug*) é frequente e, neste caso, a atitude natural é descobrir "como eu posso consertar", valorizando o próprio protagonismo. Este processo de depuração, procurando erros em uma estrutura de abstrações hierárquicas, promove o que ele chama de pensamento procedural. Papert refere que críticos do uso de computadores alegam que, ao trabalhar muito tempo com computadores, a pessoa acaba por pensar como um computador. No entanto "pensar como um computador" é mais uma maneira de pensar e esta visão procedural pode influenciar positivamente a forma como as pessoas resolvem problemas em outras circunstâncias (Papert, 1980).

Em 2010 nos Estados Unidos, o National Research Council promoveu um encontro para debater o pensamento computacional. No relatório lê-se que os conceitos de ciência da computação como abstração, algoritmo, processo, máquina de estado, especificação de tarefas, correção formal de soluções, aprendizagem de máquina, recursão, canalização, otimização etc. também são encontrados em outras ciências. Por exemplo, os físicos têm usado de abstração e de modelagem há séculos; a logística e ciências da gestão têm estudado extensivamente o agendamento de tarefas; e as noções de conciliação e equilíbrio são utilizadas entre economistas e engenheiros. No entanto, a ciência da computação fornece uma base para uma estrutura unificada e linguagem para discutir tais noções de forma explícita. Embora neste encontro não se tenha chegado a um consenso sobre a definição de pensamento computacional, várias abordagens foram registradas para descrever este conceito (National Research Council, 2010). Algumas dessas visões são apresentadas a seguir.

Um grupo representado por David Moursund argumenta que pensamento computacional é o mesmo que Papert chama de pensamento processual em seu livro *Mindstorms*. Pensamento processual inclui o desenvolvimento, representação, procedimentos de testes e depuração. Um procedimento eficaz é um conjunto de instruções que podem ser executadas mecanicamente. Dor Abrahamson tem uma visão ligada à semiótica. Ele entende o pensamento computacional como o uso de sistemas de símbolos computadorizados para articular conhecimento explícito e objetivar conhecimento tácito, manifestando este conhecimento de forma computacional e gerenciando os produtos emergentes desses esforços intelectuais. Edward Fox, Brian Blake, Uri Wilensky e Janet Kolodner associam a o pensamento computacional com a resolução de problemas e a modelagem. Ele propicia o tratamento e a manipulação de abstrações com o propósito de resolver problemas. Segundo Fox, pensamento computacional é o que as pessoas fazem quando abordam o mundo, isto é, as pessoas criam seus enquadramentos, paradigma, filosofia, linguagens, modelos e meta-modelos para explicar o mundo e resolver problemas. Dentre estas abordagens está a representação digital e suas manipulações. Pensamento computacional deve incluir a visualização, modelagem e meta-modelagem. Robert Constable evita definições estáticas. Segundo ele, mais do que uma lista de competências e processos de pensamentos, o pensamento computacional é uma lista aberta e crescente que reflete a dinâmica da natureza humana, do conhecimento e da tecnologia. Constable afirma que o que há de especialmente relevante no pensamento computacional é que os computadores podem executar nossos pensamentos computacionais e se tornar parceiros e colaboradores no processo de descoberta. Roy Pea, Ursula Wolz, Mitchel Resnick, Eric Roberts e Alan Kay discerniram sobre a programação para o desenvolvimento do pensamento computacional. Argumentaram que uma linguagem de programação é útil como forma de expressão, mas que pensamento computacional é mais do que a programação. Resnick adverte que isso não diminui a importância de se aprender a programar, assim como a escrita é menos do que se alfabetizar, a escrita ocupa um papel de destaque para se alfabetizar, analogamente a linguagem de programação ocupam papel de destaque para se atingir o pensamento computacional. A escrita e a linguagem de programação são porta de entrada para objetivos maiores. . Alan Kay afirma que, diferente da língua falada, a escrita, a matemática, a linguagem lógica e a linguagem de programação devem ser ensinadas e praticadas, necessitando um esforço bem maior para a aprendizagem do que da língua falada. Robert Constable, Peter Lee, Andy diSessa, Owen Astrachan e Ken Kahn veem o pensamento computacional como sendo a automação das

abstrações. Segundo eles, apesar da física e da matemática também se ocuparem com as abstrações para gerir e controlar a complexidade, métodos computacionais acrescentam a automação para atingir este fim. Na visão de Peter Lee o pensamento computacional amplia a inteligência das pessoas por meio da automação, resolução de problemas e gestão da complexidade. Modelagem e simulação permitem a automação da gestão da complexidade (National Research Council, 2010).

De uma forma geral podemos reduzir a quatro as visões sobre pensamento computacional predominantes: (1) Forma de Ver e Conhecer – é a corrente que define pensamento computacional como a capacidade de ver e conhecer o mundo como peças ou objetos que se relacionam e interagem. Nesse aspecto, a complexidade do mundo reside em um modelo no qual podemos decompor as interações entre objetos mais simples. (2) Maneiras ou Capacidades de Realizar – é a corrente influenciada pelo construcionismo de Papert que dá ênfase à criação de mundos simulados usando princípios de modelagem. Para esta corrente o processo de construção da simulação é o foco do pensamento computacional. (3) Método de Investigação – é a corrente que, assim como a anterior, enfatiza a simulação, mas com o objetivo de responder a uma questão de investigação, explorando a possibilidade que poderá ou poderia existir. (4) Forma de Colaboração – é a corrente que tem o pensamento computacional como estrutura para colaboração facilitada por ferramentas computacionais. A colaboração em rede e sincronização são questões importantes para este grupo (National Research Council, 2011).

2.2 Por que ensinar Pensamento Computacional?

Em 2012 a Royal Society publicou o relatório *Shut down or restart? The way forward for computing in UK schools*, no qual Steve Furber argumenta que ciência da computação influencia e dá suporte a muitas outras ciências e é necessária para o desenvolvimento de um país e tem impacto na sua economia. É importante para o indivíduo na medida em que desenvolve o pensamento superior e competências analíticas. Para as Tecnologias de Informação, o argumento é que é necessária para os dias atuais devido à ubiquidade dos computadores e seus aplicativos e também melhora a capacidade da criatividade e da expressão (Furber, 2012).

A matemática, a engenharia e as ciências naturais deram origem à ciência da computação. A mediação dos novos mecanismos desenvolvidos pela ciência da computação destacou uma forma de ver o mundo, desenvolvendo uma nova ferramenta cognitiva: o

pensamento computacional. A programação de computadores é uma forma de pôr em prática o pensamento computacional, mas esta forma de pensamento pode ser aplicada em diversas áreas como a biologia, a economia ou ciências sociais. O pensamento computacional também ajuda a perceber a sociedade digital dos nossos dias e pratica competências necessárias no século XXI, como a resolução de problemas, a criatividade e o pensamento lógico (Balanskat & Engelhart, 2014).

Em 2003 a Association for Computing Machinery (ACM), publicou um documento chamado *A Model Curriculum for K–12 Computer Science* (Tucker et al., 2003) onde se encontram alguns argumentos para que a programação de computadores e outros elementos da ciência da computação sejam ensinados nas escolas.

O conhecimento do Pensamento Computacional é uma demanda da sociedade atual, estimulando níveis cognitivos superiores de pensamento e é um fator importante para o desenvolvimento econômico de um pais em diversas áreas de conhecimento.

2.2.1 Pensamento Computacional e criatividade

O pensamento computacional dá oportunidade para o aluno se expressar e ser criativo. Assim como se ensina música, pintura e expressão escrita nas escolas, a programação também necessita de experimentação, de aprender fazendo. Da mesma forma que não imaginamos que todos os alunos serão músicos, pintores ou escritores, não podemos supor que todos venham a ser programadores profissionais, mas estes princípios básicos dão a oportunidade aos alunos para serem criativos e apreciarem a música, a pintura, a escrita e a programação como alternativas de expressão ao longo da vida (Tucker et al., 2003).

2.2.2 Pensamento Computacional e colaboração

O pensamento computacional possibilita o desenvolvimento de projetos multidisciplinares e a resolução de problemas colaborativamente nas mais diversas áreas de conhecimento. Artistas, filósofos, *designers* e cientistas de todas as áreas podem colaborar na atividade intensamente criativa de resolução de problemas. O aluno de programação trabalha com outras áreas de conhecimento de modo a representar o problema de forma clara e não ambígua sendo assim possível ser representado e tratado por um computador. Esta capacidade de expressar o problema de forma que o computador possa resolver caracteriza o pensamento computacional. Neste processo, pessoas diferentes veem aspectos diferentes e trabalham em conjunto,

reconhecendo e contornando as limitações de cada um. O mesmo ciclo de descrição-execução-reflexão-depuração, utilizado para o desenvolvimento de um programa, pode ser estendido a outros projetos maiores de diversas áreas do conhecimento. A programação permite este caráter interdisciplinar (Tucker et al., 2003).

2.2.3 O professor de Matemática e sua formação tecnológica

Pensamento computacional também promove o que se chama de fluência em tecnologia, isto é, a capacidade de se utilizar o pensamento algorítmico para se resolver problemas de forma criativa, sendo assim mais abrangente do que o letramento digital, que seria a utilização da tecnologia atual em um determinado escopo. Os algoritmos podem ser criados pelos alunos para uma variada gama de utilizações, dentro e fora da matemática. O futuro professor de matemática deve ter conhecimentos básicos sobre tecnologia para que possibilite a interdisciplinaridade, como já acontece com a matemática e outras disciplinas. A ACM sustenta que a ciência da computação deveria ser ensinada como parte das ciências naturais, assim como matemática, física, química e biologia. As razões para isso, defendem Tucker e seus colegas (Tucker et al., 2003), são que a ciência da computação é importante intelectualmente, dá acesso a múltiplas carreiras profissionais, desenvolve a capacidade de resolução de problemas, dá suporte a outras ciências, ajuda a se relacionar com outras ciências e motiva os alunos na fase escolar.

Ao estudar os currículos dos cursos de ciências (química, física e biologia) e matemática no estado de São Paulo, Bezerra e Silveira afirmam que estes objetivos da ACM não são comtemplados nos cursos de ciências em geral e que os cursos de licenciatura em matemática ensinam informática com uma abordagem utilitária de *software*, raramente tratando do assunto no contexto da Informática na Educação (Bezerra & Silveira, 2011).

Diretrizes Curriculares para os cursos de licenciatura em matemática preveem que o licenciando deve adquirir familiaridade com o uso do computador e outras tecnologias, como instrumentos de trabalho, incentivando-se a sua utilização para o ensino de matemática, com destaque para a formulação e resolução de problemas (Gatti & Nunes, 2013). No entanto, Gatti e Nunes analisaram os currículos de 31 cursos de licenciatura em matemática como amostra de todas as regiões do Brasil e verificam que apenas um deles não possui uma disciplina isolada para trabalhar com conceitos ligados à computação. Quando se trata de uso da informática para a educação, esta é referida claramente em apenas nove dos cursos. Três dos cursos

apresentam várias disciplinas com ementas fazendo referência às novas tecnologias de informação e comunicação. Observa-se, no entanto, que as ementas mostram mais uma discussão sobre a utilização dessas tecnologias do que a sua aplicação propriamente dita. Gatti e Nunes questionam se a forma como esse conhecimento vem sendo ministrado favorece a utilização das novas tecnologias nas práticas de ensino dos futuros professores, ou seja, se disciplinas que apenas discutem teoricamente a informática no ensino e que fornecem fundamentos da computação são suficientes para uma futura prática docente com utilização das novas tecnologias (Gatti & Nunes, 2013).

2.2.4 Pensamento Computacional e carreira profissional

O conhecimento do pensamento computacional desperta para várias profissões diretamente relacionadas, como programador, analista de sistemas, *web designer* ou gestor de banco de dados. Mas também abre portas para resolver problemas em outras áreas como a biologia, a criação de novos medicamentos, o desenvolvimento de projetos sociais e soluções inovadoras para o problema climático. Estes novos problemas do século XXI vão gerar carreiras no futuro, que hoje não existem, integrando diversas áreas de conhecimento, sendo o uso da tecnologia e da computação o elo que vai ligar estas áreas para enfrentar os problemas que virão (Tucker et al., 2003).

2.3 Ciência, tecnologia e letramento: delimitando fronteiras

Barcelos e Silveira definem o início da computação moderna a partir da publicação do artigo "On computable numbers, with an application to the Entscheidungsproblem" por Alan Turing, em 1936, onde é feita uma prova matemática da viabilidade da realização de cálculos numéricos a partir de uma máquina conceitual, posteriormente denominada Máquina Universal de Turing. Com este artigo são lançadas as bases estruturais da computação moderna (Barcelos & Silveira, 2012). Além da matemática, outras áreas foram incorporadas no corpo de conhecimento da computação para viabilizar os computadores eletrônicos, compiladores e linguagens de programação. A ciência da computação se baseia nas ciências naturais, na engenharia e na matemática. Barcelos e Silveira afirmam que o princípio da experimentação do método científico é utilizado no desenvolvimento de algoritmos heurísticos ou na definição de modelos para o desenvolvimento de *software*. O projeto de desenvolvimento de *software* envolve atividades de engenharia, ou seja, da aplicação da técnica. A matemática, com sua

representação simbólica e sistemas de dedução fundamentados em axiomas, é base para o estudo da complexidade de algoritmos e da análise numérica. A ciência da computação também considera o pensamento computacional como um mecanismo chave do pensamento que é possível ser aplicado em diversas áreas, como a biologia, as ciências sociais a economia e outras tantas, permitindo a análise de uma quantidade de dados impensável sem os recursos da computação (Barcelos & Silveira, 2012)

O relatório da Royal Society (Furber, 2012), apoiado pelo governo do Reino Unido, defende mudanças no ensino da Tecnologias de Informação e Comunicação (TIC) para uso nas escolas. De acordo com este relatório, o currículo dava margem a uma variedade de interpretações que levou a que fossem ministrados como educação em TIC uma série de assuntos, capacidades e competências, indo desde ser capaz de usar um *mouse* à construção de um computador, escrever um novo *software* ou a compreensão de princípios abstratos relativos à computação. Desde o início, o relatório separa a panaceia identificada por TIC em três áreas distintas denominadas de ciência da computação, tecnologias de informação e letramento digital.

A ciência da computação seria uma disciplina comparada com a matemática ou a física que, como tal, possui um corpo de conhecimento, um conjunto de técnicas e métodos rigorosos, uma maneira de pensar (pensamento computacional) e de trabalhar, um conjunto estável de conceitos e uma existência que é independente das tecnologias. Ciência da computação inclui conceitos como programação, algoritmos, estruturas de dados, arquitetura e comunicação. Entre os métodos utilizados pela ciência da computação se destacam a modelagem, a decomposição de dados e de problemas em subproblemas, a generalização de casos particulares para reutilização e o desenvolvimento de programas incluindo o seu desenho, codificação, teste, reflexão e depuração. Os métodos e o pensamento computacional se encontram em outras áreas de conhecimento como a engenharia e as ciências em geral. Em especial, na matemática encontram-se muitos aspetos comuns ao pensamento computacional. O estudo de algoritmos, considerado um elemento de matemática discreta, e o uso de tecnologia para ensinar modelagem matemática são comuns à ciência da computação e à matemática, tais como a abordagem lógica e rigorosa (Furber, 2012).

As tecnologias de informação estão relacionadas com a aplicação de sistemas de computador e a montagem, instalação, configuração e uso de *software* pré-existente para atender às necessidades dos usuários. As tecnologias de informação envolvem o uso de *software* para armazenar e manipular dados; a criação e apresentação de informações; o desenho e

configuração de sistemas; o planejamento de projetos e de sistemas informáticos; a segurança e etiqueta *online*; a reunião das questões sociais, económicas, éticas, morais, legais e políticas levantadas pelo uso generalizado da tecnologia em casa, no trabalho e para o lazer. Apesar da rápida evolução tecnológica, as tecnologias de informação continuam se ocupando da manipulação e comunicação de informações, criação e desenho de recursos, avaliação e detecção de adequação à finalidade, consciência acerca das implicações do uso difundido da tecnologia na sociedade (Furber, 2012). Alguns autores restringem as tecnologias de informação aos aspectos técnicos, enquanto os associados ao fluxo de trabalho, pessoas e informações envolvidas são tratados como Sistemas de Informação (Laurindo, Shimizu, Carvalho, & Rabechini Jr, 2001).

O letramento digital é uma competência fundamental para o aprendizado de diversos assuntos, inclusive ciência da computação e tecnologias de informação. O letramento digital capacita o aluno para utilizar o computador com confiança e eficácia ao usar aplicações de escritório, como editor de texto, planilha e apresentações; permite o uso da Internet através de navegação, pesquisa, criação de conteúdo, comunicação e colaboração via *e-mail*, redes sociais e fóruns de discussão; permite o uso de aplicativos de criação como editores de fotos, vídeo e áudio (Furber, 2012). Souza (2007) destaca que existem dois tipos de concepções para o letramento digital. Uma mais restrita associada ao uso meramente instrumental das tecnologias e outra que considera o contexto sócio-cultural, histórico e político. Ou seja, letramento digital seria uma prática cultural socialmente constituída e equivalente a aprender outra língua, dominando seus símbolos e possibilidades de comunicação e utilizando-os de modo crítico.

2.4 Incentivar o desenvolvimento do Pensamento Computacional

O relatório da Royal Society indica a necessidade de formação dos professores que hoje estão atuando nas escolas. Devido à perspectiva ampla e evasiva que hoje existe no currículo do Reino Unido e à pouca formação dos professores com relação à tecnologia, o ensino tem focado tarefas rotineiras de letramento digital. Muitas vezes este ensino é entregue ao aluno nativo digital que demonstra mais competência que seus professores. Isso faz com que os alunos rotulem TIC como uma disciplina chata, o que explicaria o desinteresse no Reino Unido por formações de nível superior como ciência da computação e tecnologias de informação. É preciso

investir em formação inicial e continuada de professores, com ênfase no ensino da ciência da computação, pensamento computacional e noções de tecnologias de informação para mudar este quadro. Professores de diversas disciplinas, inclusive de matemática, devem ser formados com estas competências (Furber, 2012).

Nos Estados Unidos, em 2011, no ano seguinte ao encontro promovido pelo National Research Council que debateu o pensamento computacional, outro encontro foi promovido para discutir como seria um currículo que promovesse o pensamento computacional (Furber, 2012). Algumas das conclusões mencionadas estão descritas a seguir.

Mitchel Resnick, Jill Denner e Idit Caperton abordam a parte social da programação. Segundo Resnick a capacidade de criação e cooperação social costuma ser negligenciada pelas escolas, mas criar, construir e inventar soluções para os problemas são fundamentais para o pensamento computacional. Eles referem que, com a cooperação, os alunos se tornam mais produtivos e motivados. Aldred Aho afirma que engajar os alunos do ensino médio em projetos criativos de programação motiva os alunos a seguirem a carreira em ciência da computação. Segundo Aho é comum ouvir relatos nos quais os alunos indicam que o mais importante que eles aprenderam não foi a programação, mas a interação e a diversão que eles tiveram em realizar os projetos. Aho cita Caperton para rejeitar a hipótese de que computadores e tecnologias de informação desumanizam as pessoas: o uso de tecnologias de informação de forma adequada "envolve as pessoas, envolve suas almas, sua paixão e sua produtividade e cuidados de pessoas" (National Research Council, 2011).

John Jungck e Michelle Williams falam da mudança do perfil do professor perante as práticas do pensamento computacional. Segundo eles, o professor tem dificuldades em colocar os interesses dos alunos em destaque, isto é, dar liberdade aos alunos para seguirem seus interesses. Nos casos em que os alunos resolvem problemas de seu interesse, os professores perdem muito do controle que normalmente têm do processo e, por vezes, se sentem desconfortáveis com isso. Dar suporte em um processo de colaboração autodirecionada exige do professor uma capacidade de diagnosticar as dificuldades e apresentar caminhos para superá-las ao invés de fornecer soluções. Os professores precisam se tornar proficientes em ensinar pensamento computacional. Eles precisam de apoio para assumir este novo papel com o qual não estão familiarizados. Os professores que se dispõem a assumir esta nova postura se tornam mais proficientes no uso das tecnologias e em fazer perguntas provocativas. As perguntas

provocativas guiam os alunos e estimulam a pesquisa e a resolução de problemas, sem prescrever receitas para as soluções (National Research Council, 2011).

Resnick afirma que os pensadores computacionais devem ser capazes de utilizar mídias computacionais para criar, construir e inventar soluções para os problemas. Ele argumenta que quando se ensina uma língua não se ensina apenas linguística e gramática, mas permite-se a expressão dos alunos. Algo semelhante deve acontecer para os pensadores computacionais, argumentando que incorporar abstração em atividades concretas poderá ser mais útil e inteligível para os alunos. Para isso o aluno deve desenvolver novos conceitos e capacidades. No entanto, Resnick enfatiza que as aulas privilegiam mais os conceitos do que as competências. Os conceitos computacionais incluem o conceito de condicional, processo, sincronismo e recursividade. A capacidade de realizar projetos lida com competências como prototipação, abstração, modelagem e depuração. Capacidades sócio-cooperativas incluem compartilhar, colaborar, *remixar* e a colaboração em massa. Estas capacidades sócio-cooperativas se tornam cada vez mais importantes à medida que as novas tecnologias da computação e de rede abrem novas possibilidades de cooperação largamente difundida (National Research Council, 2011).

2.4.1 Ensino de programação integrado no currículo

Embora o pensamento computacional seja mais do que a programação de computadores, aprender uma linguagem de programação contribui em muito para o entendimento e aplicação do pensamento computacional (National Research Council, 2011).

A European SchoolNet fez uma pesquisa em 20 países europeus e Israel, para verificar a tendência de inclusão de programação de computadores nas escolas (Balanskat & Engelhart, 2014). Desse estudo, destaco aqui alguns itens que, a meu ver, contribuem para este trabalho. No relatório citado, o termo codificação é utilizado como sinônimo de programação de computadores.

2.4.1.1 Codificação no currículo: uma perspectiva internacional

Em 16 países a codificação faz parte do currículo em nível nacional, regional ou local: Áustria, Bulgária, República Checa, Dinamarca, Estónia, França, Hungria, Irlanda, Israel, Lituânia, Malta, Espanha, Polónia, Portugal, Eslováquia e Reino Unido (Inglaterra). Em dois países, Finlândia e na parte flamenga da Bélgica (Flandres), há planos para integrar a codificação no currículo. Em três países, na parte francófona da Bélgica (Valónia), Holanda e

Noruega a codificação ainda não está integrada e, atualmente, não há planos para tal (Balanskat & Engelhart, 2014).

2.4.1.2 Por que integrar codificação no currículo?

A maioria dos países tem como objetivo desenvolver as competências dos alunos de pensamento lógico (15 países) e competências de resolução de problemas (14 países), visando corresponder às competências necessárias para o século XXI. Mais da metade dos países (11 países) tem o foco no desenvolvimento da competência de codificação. Atrair mais estudantes para estudar ciências da computação também é uma justificativa para 11 países. O objetivo de fomentar a empregabilidade no sector é fundamental para apenas oito países (Balanskat & Engelhart, 2014).

2.4.1.3 Codificação como disciplina ou integração transversal?

Doze países já estabeleceram uma disciplina específica para o ensino de codificação ou computação por iniciativa regional ou da própria escola. Outros 13 países integram a codificação em um curso geral de Tecnologias de Informação e Comunicação (TIC) ou tecnologia. Em sete deles, sem depender de um empenho nacional, mas por uma iniciativa regional de entidades regionais ou da própria escola. Cada vez mais a codificação ou computação é integrada em outras disciplinas, principalmente matemática, em uma abordagem transversal, como, por exemplo, na Dinamarca, Estónia, Finlândia, Eslováquia, Espanha e França. Finlândia será o primeiro país a introduzir a codificação em uma abordagem puramente transdisciplinar (Balanskat & Engelhart, 2014).

2.5 Pensamento Computacional e ensino de Matemática

Existem conhecimentos partilhados entre o pensamento computacional e a matemática, como o uso de abstrações, algoritmos e lógica (Barcelos & Silveira, 2012; Furber, 2012). Deste modo os professores de matemática podem ajudar no ensino do pensamento computacional e o pensamento computacional pode ser uma aplicação e uma motivação para o ensino de matemática, desenvolvendo competências comuns às duas áreas de conhecimento.

Em seguida são apresentadas algumas possibilidades explorando as áreas em comum do pensamento computacional e das ciências, tecnologia, engenharia e matemática (STEM, do

inglês: Science, Technology, Engineering and Mathematics) e a relação entre o pensamento computacional e os Parâmetros Curriculares Nacionais do Brasil (PCN).

2.5.1 Pensamento Computacional e STEM

David Weintrop e colegas (Weintrop et al., 2016) propõem uma abordagem partindo das competências necessárias para atingir o pensamento computacional. Foi realizada uma pesquisa que originou uma taxonomia na qual o pensamento computacional é definido a partir das capacidades que a compõem levando em conta as recomendações dos especialistas e dos professores que estão ministrando suas aulas nas escolas. Apesar da falta de consenso entre os especialistas, alguns aspectos como a abstração e a criação de algoritmos convergem para definir o que é pensamento computacional. A pesquisa teve como alvo o ensino das áreas de ciência, tecnologia, engenharia e matemática, conhecidas pela sigla inglesa STEM, focando no consenso dos especialistas, em uma postura dedutiva (de cima para baixo) e a partir das práticas dos profissionais STEM ligados à educação escolar, em uma postura indutiva (de baixo para cima). Nessa taxonomia as competências exigidas tanto para o pensamento computacional como para as áreas STEM são classificadas em 4 categorias: (1) Competências com Dados e Informações, (2) Competências de Simulação e Modelagem, (3) Competências de Resolução de Problemas e (4) Competências do Pensamento Sistêmico. Weintrop e seus colegas defendem que o pensamento computacional deve ser ensinado no através dos profissionais STEM.

Vejamos, com mais detalhe, cada uma das quatro categorias taxonômicas. As Competências com Dados e Informações se referem ao uso de dados em STEM que aparecem de formas variadas com diferentes propósitos. Os dados possibilitam questões interessantes e desafiadoras. Para se valer dos dados, os alunos usam competências comuns tanto para STEM como para o pensamento computacional. Esta categoria é formada pelas competências de: (a) Coletar dados; (b) Criar dados; (c) Manipular dados; (d) Analisar dados; (e) Visualizar Dados (Weintrop et al., 2016).

Competências de Simulação se refere ao uso de modelo em STEM. Com o computador torna-se possível realizar modelos dinâmicos do mundo real que seria muito caro, muito perigoso, muito difícil ou impossível de realizar de outra forma. Os modelos computacionais, assim como os demais modelos STEM, são simulações do mundo real, estando sujeitos a limitações. Esta categoria é formada pelas competências de: (a) Usar modelos computacionais para entender um conceito; (b) Entender como e por que os modelos computacionais

funcionam; (c) Avaliar modelos computacionais; (d) Usar modelos computacionais para achar e testar soluções; (e) Construir e/ou expandir modelos computacionais (Weintrop et al., 2016).

Competências de Resolução de Problemas se refere à superação de desafios trazidos pelas STEM utilizando estratégias computacionais. Deve-se ter a habilidade de se expressar de forma que computadores possam interpretar, executar e investigar fenômenos STEM. Esta categoria é formada pelas competências de: (a) Detectar e resolver erros; (b) Programar computadores; (c) Escolher ferramentas computacionais eficazes; (d) Avaliar diferentes abordagens e soluções para um problema; (e) Desenvolver soluções computacionais modulares; (f) Usar estratégias de resolução de problemas; (g) Criar Abstrações (Weintrop et al., 2016).

Competências do Pensamento Sistêmico se refere à habilidade de identificar elementos que interagem mutuamente constituindo um sistema. Estes elementos podem ser qualquer coisa desde peças mecânicas no motor de um carro até elementos químicos em uma solução. Esta categoria é formada pelas competências de: (a) Investigar um sistema como um todo; (b) Compreender as relações internas de um sistema; (c) Pensar em diferentes níveis; (d) Visualizar Sistemas; (e) Identificar, entender e gerenciar a complexidade (Weintrop et al., 2016).

2.5.2 Pensamento Computacional e parâmetros curriculares Brasileiros

Barcelos e Silveira (2012) fizeram o mapeamento entre as competências da área de matemática previstas nos Parâmetros Curriculares Nacionais do Brasil (PCN) e aquelas desenvolvidas em atividades didáticas envolvendo o pensamento computacional descritas na literatura. Os PCN, ao se referirem à necessidade do ensino de matemática estar associado à tecnologia, fazem também referência às competências típicas do pensamento computacional:

> Esse impacto da tecnologia, cujo instrumento mais relevante é hoje o computador, exigirá do ensino de matemática um redirecionamento sob uma perspectiva curricular que favoreça o desenvolvimento de competências e procedimentos com os quais o indivíduo possa se reconhecer e se orientar nesse mundo do conhecimento em constante movimento.
> Para isso, competências como selecionar informações, analisar as informações obtidas e, a partir disso, tomar decisões exigirão linguagem, procedimentos e formas de pensar matemáticos que devem ser desenvolvidos ao longo do Ensino Médio, bem como a capacidade de avaliar limites, possibilidades e adequação das tecnologias em diferentes situações. (Ministério da Educação, 2000, p. 41)

Este trabalho apontou que, nos PCN, as atividades que podem contribuir para o desenvolvimento do pensamento computacional se localizam sob três competências, a saber: (1) Articulação de símbolos; (2) Identificação de padrões e regularidades; (3) Construção de

modelos representativos e explicativos. Portanto as atividades de programação que atendam a estas competências matemáticas ajudam no desenvolvimento do pensamento computacional (Barcelos & Silveira, 2012).

A articulação de símbolos é uma capacidade esperada do aluno de matemática. Espera-se que ele seja capaz de traduzir uma situação-problema em uma linguagem discursiva e depois em uma representação matemática através de gráficos, tabelas ou fórmulas, por exemplo. Neste contexto a linguagem de programação possibilita mais uma representação em forma de algoritmo. Espera-se também que o aluno seja capaz de transformar uma representação em outra, como por exemplo, transformar os dados de uma tabela em um gráfico. Neste caso a forma sequencial do algoritmo, representando um procedimento passo a passo, aproxima esta representação da linguagem discursiva. De modo que, ao representar um problema na forma algorítmica, possibilitar-se-ia um degrau de cognição intermediário entre a linguagem discursiva e a linguagem algébrica. A programação também possibilita conceitos como "variável" e "domínio" que também existem na linguagem algébrica, mas o caráter dinâmico da programação pode ajudar a entender estes conceitos, na medida em que permite a experimentação. A experimentação da programação permite experimentações matemáticas, uma vez que a representação algorítmica é mais uma das representações simbólicas usadas pela matemática. O aluno pode experimentar variar o comportamento de uma reta, representada por um algoritmo, experimentando diferentes valores para o coeficiente angular (a) e para o termo independente (b) em uma reta ($y = a \cdot x + b$), como destacam Barcelos e Silveira (2012). Esta experimentação leva o aluno a levantar conjecturas a respeito do ente matemático realizando o que Vergnaud chama de Teorema-em-Ação, levando o aluno ao verdadeiro fazer matemático (Barreto, Camelo, Fernandes, Pequeno, & Filho, 2009).

Identificação de padrões e regularidades também é uma capacidade valorizada nos parâmetros curriculares e no pensamento computacional. Deseja-se que, ao se observar uma situação-problema o aluno seja capaz de estabelecer hipóteses, tirar conclusões, generalizar situações e abstrair regularidades. Por exemplo, em uma sequência de números como (2, 5, 8, 11, 14,), qual é a regra que determina o próximo número da sequência? Ou então, no algoritmo da Figura 1, qual a regra que relaciona o número de lados de um polígono com o ângulo de giro do traçado?

A programação possibilita a identificação de padrões e regularidades como sugerem os PCN.

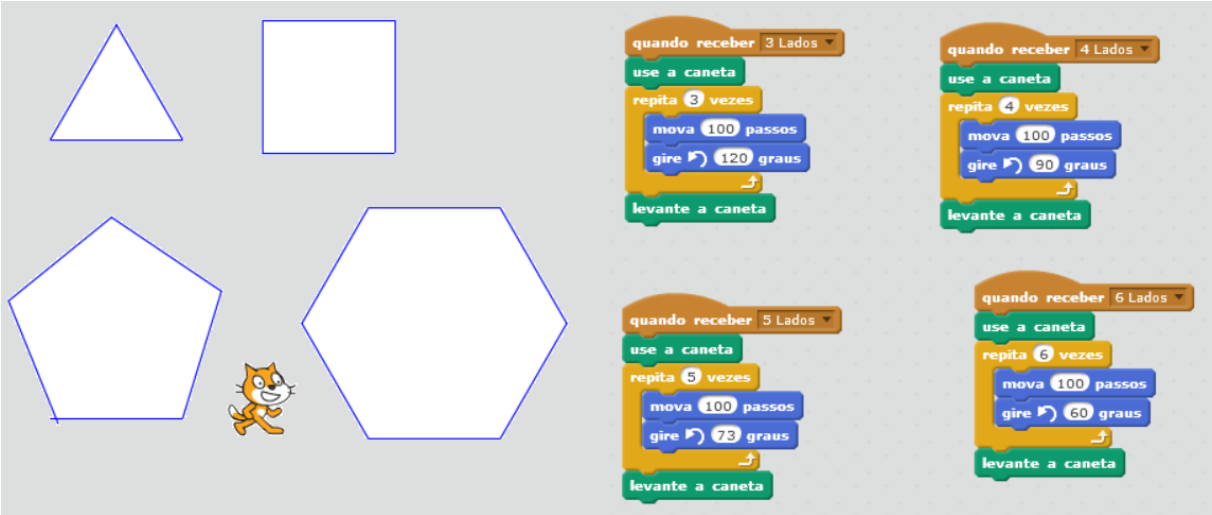


Figura 1 - Traçado de Polígonos

A construção de modelos representativos e explicativos, solicitada pelos parâmetros curriculares se ajusta à modelagem e simulação de fenômenos requerida pelo pensamento computacional (Barcelos & Silveira, 2012). Segundo os parâmetros curriculares, as situações devem estar associadas ao cotidiano do aluno, tais como: cálculos de lucro e prejuízo envolvendo gráficos, estimação das intenções de voto em uma campanha eleitoral envolvendo conceitos de estatística e probabilidade, entre outros. A construção de modelos pelos alunos com assuntos do seu próprio interesse caracteriza a tendência do ensino de matemática denominada de modelagem matemática. Barcelos e Silveira citam como exemplo uma atividade em que alunos produzem e testam um modelo de contágio de doenças considerando a quantidade de alunos e a disposição física dos ambientes da sua escola. O modelo é testado, baseado na criação de agentes computacionais, utilizando uma ferramenta com uma linguagem de programação com blocos de comandos, semelhante ao Scratch. Tais ferramentas permitem a criação de modelos para explorar fenômenos razoavelmente complexos. Conceitos matemáticos associados ao modelo, como a probabilidade de contágio e tempo de incubação, podem ser manipulados dinamicamente através da alteração de parâmetros na própria ferramenta (Barcelos & Silveira, 2012).

Como se pode notar, os PCN recomendam para o ensino de matemática práticas que ao mesmo tempo desenvolvem o pensamento matemático e o pensamento computacional, sendo a programação uma opção para praticar estes dois tipos de pensamento. A linguagem de programação deve ser expressiva e não ambígua a ponto de ser interpretada por uma máquina

de inferências. Observe como o discurso dos PCN (Ministério da Educação, 2002) pode ser estendido à programação e ao pensamento computacional:

> Ao final do ensino médio, espera-se que os alunos saibam usar a matemática para resolver problemas práticos do quotidiano; para modelar fenômenos em outras áreas do conhecimento; compreendam que a matemática é uma ciência com características próprias, que se organiza via teoremas e demonstrações; percebam a matemática como um conhecimento social e historicamente construído; saibam apreciar a importância da matemática no desenvolvimento científico e tecnológico.
> A forma de trabalhar os conteúdos deve sempre agregar um valor formativo no que diz respeito ao desenvolvimento do pensamento matemático. Isso significa colocar os alunos em um processo de aprendizagem que valorize o raciocínio matemático – nos aspectos de formular questões, perguntar-se sobre a existência de solução, estabelecer hipóteses e tirar conclusões, apresentar exemplos e contraexemplos, generalizar situações, abstrair regularidades, criar modelos, argumentar com fundamentação lógico-dedutiva. (Ministério da Educação, 2002, p. 69)

3 Ensinar, aprender, avaliar: o olhar fundamental

Para que a aprendizagem seja efetiva em sala de aula deve haver um processo que permita ao professor ensinar e ao aluno aprender. Este processo de ensino-aprendizagem é, a meu ver, um processo sinérgico e multifacetado. Por isso mesmo, apenas a justaposição de ensino e aprendizagem não é suficiente para descrever o processo. Num processo ensino-aprendizagem, acontece a aprendizagem do aluno e, por vezes, do professor atento. O produto desse processo é a aprendizagem, mas o próprio processo é por vezes chamado de aprendizagem. No dicionário Houaiss existem três significados para a palavra aprendizagem: "(1) Ato ou efeito de aprender. (2) O tempo que dura tal aprendizagem. (3) O exercício ou prática inicial da matéria aprendida" (Houaiss, 2001). No primeiro sentido vemos aprendizagem como produto do processo e nos dois seguintes vemos aprendizagem denominando o processo ensino-aprendizagem. Neste trabalho estou interessado no processo, naquilo que acontece em sala de aula e em torno dela, e como o professor pode contribuir para ele. Portanto, o sentido de aprendizagem que usarei aqui se referirá prioritariamente ao processo de ensino-aprendizagem, na tentativa de esclarecer como as contribuições aqui expostas atuam de maneira multifacetada e sinergicamente para aluno e para o professor.

A pedagogia da aprendizagem tem somado vertentes ao longo da história. Rapidamente eu gostaria de comentar as seguintes concepções de aprendizagem: aprender é lembrar, aprender é mudar o comportamento, aprender é processar informações e aprender é interagir (Roseira, 2004).

Aprender é Lembrar – Nesta concepção o aluno busca na sua memória os conteúdos absorvidos nas situações de ensino. Paulo Freire chama de educação bancária (Freire, 1987), fazendo analogia com os depósitos bancários e os depósitos de conhecimento no aluno. Outros se referem a educação *baldista*, em analogia do aluno como sendo um balde a ser preenchido (Santos, 2002). Neste processo, os pacotes de conhecimentos são inquestionáveis e se espera nas avaliações uma única resposta correta. Ao aluno cabe memorizar e reproduzir.

Aprender é Mudar o Comportamento – Esta concepção é também chamada de behaviorismo. Tem a visão do organismo que aprende como sendo uma caixa preta que está sujeito a estímulos que causam um comportamento como efeito, podendo esses comportamentos ser estabelecidos através de estímulos, reforçando os comportamentos desejados. Esta concepção, muito em voga nos anos setenta do século passado, levou a

propostas pedagógicas como o Ensino Programado e a Aprendizagem por Objetivos (Roseira, 2004). O professor estabelece um objetivo final e os comportamentos que vão ser avaliados para confirmar este objetivo. O objetivo é quebrado em vários sub-objetivos com pequenas dificuldades, evitando que o aluno erre. O aluno é conduzido por esta sequência e pelo professor ao objetivo final (Santos, 2002). Este modelo se mostra eficiente a curto e médio prazo para o ensino de procedimentos e automatismos, porém críticas são feitas pelo excesso de condução, levando o aluno a não entender o todo: por exemplo, o saber girar o volante, pisar no freio e trocar de marcha não quer dizer que se saiba dirigir um automóvel. Outro aspecto criticado é a dependência do aluno do direcionamento, comprometendo a autonomia. Porém, diferente da concepção bancária, o professor deve considerar o tamanho dos passos rumo aos objetivos em função dos alunos, preocupando-se com uma sequência didática adequada ao seu público (Santos, 2002).

Aprender é Processar Informações – Nesta concepção assume-se que o aluno é capaz assimilar conhecimento depois de um estágio de amadurecimento lógico e racional. O aluno passa a ser o centro da aprendizagem e ao professor cabe conduzir o processo de forma significativa, ancorando os novos conhecimentos em conhecimentos prévios dos alunos, organizando e facilitando a aprendizagem. O erro passa a ser visto como parte importante do processo.

Aprender é Interagir – Esta é uma concepção que parte do pressuposto de que o indivíduo aprende por um amadurecimento biológico e pela interação com o meio. O construtivismo de Piaget foi o precursor desta concepção, alterada mais tarde pela teoria sócio-histórica de Vygotsky, que enfatizou as interações sociais vindas da cultura que o indivíduo está exposto, numa perspectiva sócio interacionista.

O avanço das teorias pedagógicas gera instrumentos que o professor pode utilizar em seu trabalho. Apesar de apostar na visão sócio-interacionista como sendo a opção mais profícua em sala de aula, não descarto as demais concepções como possibilidade de explicar e gerar bons resultados educacionais, considerando a complexidade do processo de aprendizagem.

> Sendo assim, considero que a memorização, as mudanças nas formas de agir e pensar dos alunos e o processamento de informações são importantes no processo de ensino-aprendizagem e complementam a educação como sócio-interação, porém certamente são insuficientes e incapazes de contribuírem para a compreensão da realidade, se tomados de forma isolada, fragmentada e como verdades que se sobrepõem às demais concepções. (Roseira, 2004, p. 72)

Coerente com esta tendência mais holística, ou seja, tentando o melhor de cada tendência, tentarei ensinar e motivar o aluno em vários aspectos tanto do ponto de vista social quando cognitivo.

As teorias construtivistas da educação recomendam ao professor que coloque o aluno em uma situação de desequilíbrio, isto é, em um conflito cognitivo, de modo que ele se recupere, se reequilibrando em uma outra situação confortável, ao adquirir um novo conhecimento. A resolução de problemas é, em geral, a prática adotada para diferentes objetivos. Na resolução de problemas para o ensino de matemática valoriza a discussão e o fazer matemático através do raciocínio, da argumentação e da visualização (Zulatto, 2007). O ensino de programação se apoia no ciclo natural da programação de Descrição-Execução-Reflexão-Depuração associado à resolução de problemas (Maltempi & Valente, 2000).

O objeto da investigação é o processo de aprendizagem, sendo assim, se faz necessário esclarecer o que seria a aprendizagem e como avaliar este processo. Para explorar estes temas, partirei das seguintes questões:

- Como as pessoas aprendem?
- Como se aprende matemática?
- Como se aprende a programação de computadores?
- Como avaliar o processo de aprendizagem?

3.1 Como as pessoas aprendem?

Para tentar entender como as pessoas aprendem, tive como motivação inicial o capítulo 9, intitulado “Os Torcedores de Palavras/Os Criadores de Mundos”, do livro *O Fim da Educação*, de Neil Postman. Tentei relacionar as ideias deste capítulo com outras referências que concordam com Postman e que, de certa forma, completam suas ideias. Tentei relacionar linguagem, metáforas e seus símbolos apresentadas com os modelos utilizados nas ciências e com a teoria sócio-cultural.

Postman(2007) inicia seu texto tecendo comentários sobre definição, perguntas e metáforas. O autor argumenta que estes elementos são essenciais para a educação e que devem ser mais valorizados. Sobre a definição, refere que ela é dada como definitiva, como se fosse parte do mundo natural e não uma das possíveis invenções funcionais criadas pelo ser humano. Com relação às perguntas, o autor observa que os alunos não são acostumados a fazer

perguntas. Para mim, estes dois aspectos estão relacionados ao que Paulo Freire chama de educação bancária.

No livro *Educação e Mudança*, Paulo Freire afirma que este modelo é o mais identificado como prática de ensino e menos eficiente para educar: "O professor ainda é um ser superior que ensina a ignorantes. Isto forma uma consciência bancária. O educando recebe passivamente os conhecimentos, tornando-se um depósito do educador. Educa-se para arquivar o que se deposita" (Freire, 2014, p. 20). Se o aluno é passivo, não se deve esperar dele perguntas e esta postura de superioridade do professor diante do aluno ignorante leva-o a arrogância de dizer qual é "a" definição. Esta situação me chama a atenção pela falta de perspectiva de uma interação com o aprendiz e também a falta de valorização da sua autonomia. Marco Silva (2000), em seu livro *Sala de Aula Interativa*, afirma que a interatividade é a prática válida para o século XXI, no qual nem as respostas nem as perguntas estão catalogadas. Silva comenta que mesmo as perguntas a que o aluno costuma responder são do tipo "quem descobriu o Brasil?", quando deveria ser "que aspectos você acha importantes no descobrimento do Brasil?". A diferença entre estas duas perguntas está na concepção de uma educação bancária, na primeira, enquanto a segunda se baseia na concepção de uma educação cuja interatividade é fundamental, na construção da participação cidadã, estimulando a experimentação e a criatividade.

Postman critica a concepção de metáfora como função meramente decorativa utilizada por poetas, que serve como conteúdo de aulas de literatura. "Uma metáfora não é uma ornamentação. É um órgão de percepção"; "estarei a exagerar quando afirmo que um aluno não pode compreender a que se dedica uma disciplina sem possuir algum entendimento das metáforas que lhe servem de sustentáculos? Creio que não" (Postman, 2007, p. 201). Ele sustenta que as definições, perguntas e metáforas são as bases para se construir uma "mundivisão". O conceito de mundivisão eu relaciono com o que Paulo Freire usa em *Ler o mundo* ou ao que Rubem Alves chama de "Modelos" no seu livro *Filosofia da Ciência* (Alves, 2000). Postman diz: "estou a sugerir que a criação de mundos através da língua é uma narrativa de poder, durabilidade e inspiração. É a história de como aprendemos o mundo e a nós mesmos e de como nos apresentamos a nós mesmos no mundo" (Postman, 2007, p. 201). Postman é a favor da autonomia ao defender a criação de mundos, o empoderamento da narrativa e a valorizar a inspiração.

Outro aspecto que destaco é a opção de Postman, partindo da metáfora, de utilizar a linguagem como construto dessa mundivisão, enquanto Rubem Alves diz que os modelos são baseados em conceitos.

> Um modelo é um artefato construído pelo cientista. Quando falamos em artefatos, pensamos em coisas fabricadas com o auxílio de materiais sólidos, como relógios, máquinas de moer carne, cortadores de unha, satélites artificiais. Todos são artefatos: produzidos pela arte dos homens. Para se construir um modelo fazemos uso não de materiais sólidos mas de conceitos.(Alves, 2000, p. 48)

Ao definir nossa mundivisão através da linguagem, estamos assumindo o uso de símbolos ou signos. Segundo Postman os símbolos ou signos dependem da nossa capacidade de abstração. Abstraímos ao nível neurológico, fisiológico, percepcional e verbal nossas interações com o mundo. Ocupamo-nos em selecionar, organizar e generalizar as informações para podermos atuar no mundo. Este processo suprime diferenças e identifica padrões. Nomear as coisas é um ato de abstração. Segundo Vygotsky, a linguagem não precisa ser necessariamente a fala, seja ela escrita ou oral.

> A linguagem não depende necessariamente do som. Há por exemplo a linguagem de sinais dos surdos e a leitura dos lábios, que é também interpretação de movimentos. Nas linguagens dos povos primitivos, os gestos são utilizados em paralelo com o som e desempenham um papel de certa importância. Em princípio, a linguagem não depende da natureza do material que emprega. (...) O meio de expressão não está em causa; o que importa é o uso funcional dos signos, de quaisquer signos que possam desempenhar um papel correspondente ao da linguagem humana. (Vygotsky 1989, p. 30)

Sendo assim, podemos considerar os modelos de Rubem Alves como sendo signos de uma linguagem que nos leva a uma mundivisão. O conceito de signo por sua vez é definido pelo Dicionário Priberam da Língua Portuguesa como "unidade linguística que contém um significante (forma ou imagem acústica) e um significado (conceito)" (Dicionário Priberam da Língua Portuguesa, 2013). Ora, o conceito é a base para construção dos artefatos que Rubem Alves chama de modelo, sendo assim, existe uma equivalência entre os Modelos de Rubem Alves e a Mundivisão de Postman fornecida pela linguagem.

Tanto Postman como Rubem Alves falam do conhecimento do mundo de maneira indireta, através da linguagem ou dos modelos.

> A palavra caneca, por exemplo, não denota nada que exista verdadeiramente no mundo. É um conceito, a síntese de milhões de coisas específicas que têm um aspecto e uma função similares (...) Deste modo, podemos concluir que os humanos vivem em dois

> mundos – o mundo dos eventos e das coisas e o mundo as palavras acerca dos eventos e das coisas. (Postman, 2007, p. 208)

> Nós não conhecemos a realidade. Não podemos contemplá-la face a face. Se tivéssemos uma visão direta da realidade, nosso conhecimento seria final, definitivo. Mas isto não acontece. Frequentemente, os cientistas são forçados a reconhecer que as coisas são totalmente diferentes daquilo que pensavam. Aí ocorrem as grandes revoluções na ciência. Isto não aconteceria se o conhecimento fosse visão direta do real. Ao invés de visão direta, palpites; ao invés de conhecimento certo e final, conhecimento provisório. Por quê? Porque o que temos nas mãos são os modelos. Os modelos são aquilo que conhecemos. (Alves, 1983, p. 48)

A fala de Rubem Alves explica as múltiplas definições ditas por Postman. Cada uma com uma visão ou um modelo do real. Este autor também recorre a Kant para reafirmar suas convicções:

> O que os objetos são, em si mesmos, fora da maneira como a nossa sensibilidade os recebe, permanece totalmente desconhecido para nós. Não conhecemos coisa alguma a não ser o nosso modo de perceber tais objetos – um modo que nos é peculiar e não necessariamente compartilhado por todos os seres... (Alves, 1983, p. 48)

Segundo Rubem Alves, utilizamos os modelos para ter domínio sobre o mundo:

> O homem necessita viver num mundo ordenado. Há razões biológicas e psicológicas para tal. Mas para que haja ordem é necessário *organizar o espaço e o tempo*. Você já pensou no sentido da palavra *organizar*? Ela é parenta de organismo, órgão. Organizar é transformar algo em órgão, em instrumento a serviço das necessidades de um certo organismo. Organizar o mundo é fazê-lo uma extensão do corpo, é submetê-lo a princípios de ordenação estabelecidos pelas necessidades do sujeito que organiza. A organização do espaço me permite responder à pergunta: *onde estou?* E a resposta a esta pergunta é pré-requisito para uma outra: *em que* direção caminhar? A organização do tempo, por outro lado, me diz *quanto tempo ainda tenho*, o que me resta para completar a ação programada. (Alves, 2000, p. 48)

Sendo assim, tendo o domínio sobre o mundo, mesmo que de maneira parcial, provisória e às vezes ilusória, o passo seguinte seria atuar no mundo.

Segundo Postman, o que acreditamos, ou não acreditamos, é influenciado pela maneira que a nossa língua vê o mundo. As pessoas São, Fazem e Têm. A língua cria mundivisão.

Postman refere-se a um seminário levado a cabo por I. A. Richards onde este pede a três grupos distintos que falem sobre a língua a partir de uma frase inicial. Para cada um dos grupos as frases iniciais eram diferentes. “A língua é como uma árvore...”, “ A língua é como um rio...”, “A língua é como um edifício...” Deste modo, Richards conduziu os grupos a criarem diferentes

modelos usando a língua com metáforas diferentes. A partir desse ponto, as metáforas controlavam a coerência dos discursos mantendo o modelo coerente. Este é um exemplo que Postman usa para mostrar como os símbolos geram diferentes mundos.

Rubem Alves defende que entendemos aquilo que construímos e manipulamos, como um relógio. Por isso mesmo, o universo é, por diversas vezes, comparado a um relógio cujos tempos, relacionados com o movimento da lua e do sol ou da posição das estrelas, podem ser previsíveis. Do mesmo modo o homem quer entender as colisões no universo e pode usar como modelo um jogo de bilhar no qual a bola branca colide com outra e esta, por sua vez, cai na caçapa. O jogador pode imaginar o destino desejado da bola a ser encaçapada, depois faz uma hipótese sobre o movimento da bola branca, só aí ele determina o que fará com o taco. Tendo domínio desse modelo ele irá entender as colisões do universo. O universo será uma grande mesa de bilhar.

Dependendo do modelo ou metáfora que adotamos, nossa visão e consequentemente nossa atitude diante do mundo serão diferentes. Postman levanta uma série de hipóteses explicadas por diferentes modelos ou metáforas bem conhecidas por nós, educadores: "Será a mente humana, por exemplo, como uma caverna obscura (que precisa ser iluminada)? Como um músculo (que precisa ser exercitado)? Como um recipiente (que precisa ser enchido)? Um pedaço de barro (que precisa ser moldado)? Um jardim (que precisa ser cultivado)?" (Postman, 2007, p. 201).

A linguagem e suas ferramentas simbólicas recebem influências do ambiente social. Veja, por exemplo, o caso dos esquimós. Eles distinguem neve que serve para fazer casas, neve mais lisa, neve menos escorregadia, neve que pode cair, neve sobre lago, neve sobre lago que se pode pisar, neve boa para fazer bolas e brincar, etc. (Vilela, 2002). Esta riqueza de símbolos linguísticos vai levar o esquimó a comportamentos diferentes, porque cada símbolo leva a um pensamento diferente.

Os diferentes modelos, ou conjunto de símbolos, nos levam a diferentes formas de pensar.

3.1.1 Símbolos nos ajudam a pensar

Na história da espécie humana (filogenia), sobre o pensamento e a linguagem, Vygotsky afirma que se distingue claramente uma fase pré-intelectual do desenvolvimento da linguagem e uma fase pré-linguística do desenvolvimento do pensamento. No entanto, a partir dos dois anos

de idade a evolução da linguagem e do pensamento caminham juntos. As palavras não são simplesmente faladas, mas elas servem de suporte ao pensamento. O discurso interior acontece quando internalizamos a linguagem e seus mecanismos simbólicos. Assim, não precisamos externar estes símbolos, através da fala, para compor nosso pensamento.

Postman diz que ao explicar as várias metáforas e os vários estilos literários das diversas ciências estamos contando a história da língua como ferramenta de criação. Ele cita Sócrates e Max Muller para reforçar esta relação entre linguagem e pensamento. "'Quando a mente está a pensar, está a falar consigo própria' – Sócrates". "'O pensamento não pode existir sem símbolos e nossos símbolos mais importantes são as palavras' – Max Muller" (Postman, 2007, p. 216).

Assim como usamos uma faca para cortar o pão, o símbolo é a nossa ferramenta para o pensamento abstrato (Kohl, 2015). A linguagem serve de suporte ao pensamento, sendo o pensamento verbal uma forma de pensamento. Outras formas de pensamento são possíveis, utilizando símbolos de natureza não verbal e abstrata, como os modelos matemáticos. Entre estas outras formas de mediação simbólica encontram-se os computadores e os sistemas informáticos de uma maneira geral.

Jerome Bruner foi um dos precursores da psicologia cultural, afirmando que não eram só os processos biológicos e o ambiente físico os responsáveis pelo desenvolvimento da mente, mas também os contextuais e os culturais. Mônica Correia esclarece que não há um consenso do que seja cultura, mas pode ser interpretada como os aspectos da realidade social, ou tudo o que caracteriza um povo, nação ou grupo de pessoas em uma sociedade, ou modos de organizar a vida social, ou como uma teia de significados construída e compartilhada por um grupo social. Da sociedade emerge a cultura, mas do ponto de vista cognocêntrico, muitas vezes, cultura e sociedade são tomadas como sinônimos (Correia, 2003). Bruner compartilha dessa visão de cultura como um sistema simbólico. Para ele, cultura seria um *kit* de ferramentas, técnicas e procedimentos para entender e atuar no mundo, resolvendo problemas e interagindo com outros, através de símbolos (Bruner, 2001).

Bruner rejeita a visão computacional, a qual explica a mente como uma máquina computacional capaz de tratar o conhecimento formador de informações finitas, codificadas em símbolos e não ambígua, o que contrasta com o processo do saber muitas vezes confuso e ambíguo. Para Bruner a mente trabalha com um simbolismo compartilhado que ordena e interpreta a vida. A mente humana seria moldada, ou educada, pela cultura, ao mesmo tempo em que contribui para ela, num processo de negociação dos significados de acordo com o

contexto. A mente não só assimila, mas também cria significados num processo de troca de mão dupla com a cultura. Neste processo de negociação o indivíduo cria a sua identidade. Na troca entre o indivíduo e a sociedade, a narrativa aparece como intermediária (Correia, 2003).

Para Bruner, a mente é formada por fatores biológicos, culturais e contextuais, sendo a cultura o único fator capaz de contribuir qualitativamente na estrutura que compõe a cognição. A mente é proativa, intencional e constrói suas próprias estruturas mentais, fazendo do sujeito um autor que está disposto a diálogos com outras mentes ativas, provocando a conversação e a colaboração entre as mentes. Com o intuito de promover essas transações entre mentes, tendo a narrativa como moeda de troca, os indivíduos são razoavelmente previsíveis para se fazerem entender (Correia, 2003).

As mentes se comunicam e se formam através de símbolos que têm a cultura como referência. Esta referência é dinâmica, na medida em que o coletivo de indivíduos, com seus conjuntos de símbolos e significados, altera a cultura. A interação só é possível através desses símbolos. A articulação desses símbolos é a linguagem, que vamos aqui tomar em um sentido mais amplo, sendo a linguagem tão variada quanto os subsistemas de símbolos, isto é, linguagem escrita, linguagem falada, linguagem dos surdos, linguagem audiovisual etc.. Cada uma dessas linguagens usa meios ou mídias que dão suporte aos símbolos, como a fala, o papel, o cinema e jogos eletrônicos. Este conjunto de elementos, símbolos, mídias, mentes, linguagens e cultura, agem de maneira dialética formatando diferentes maneiras de pensar.

3.1.2 Tecnologias e diferentes formas de pensar

Segundo Postman, a informática, sendo uma tecnologia, ela gera uma mundivisão. Esta visão é reforçada por Tikhomirov, discípulo de Vygotsky, ao afirmar que a informática reorganiza o pensamento. Durante os anos 1980, duas visões prevaleciam sobre como esta tecnologia afetava a sociedade. Eram as teorias da substituição e da complementação.

A teoria da substituição afirmava que o computador iria substituir o ser humano em toda esfera do trabalho intelectual na medida em que eles fossem se tornando mais rápidos e com mais capacidade de armazenamento, uma vez que poderia simular o pensamento humano. Tikhomirov alerta que esta teoria considera o conhecimento como algo fragmentado. Desta forma, pensamentos complexos poderiam ser decompostos em pensamentos mais simples e a união desses pensamentos simples levaria à complexidade sem perda de informações. Esta é a visão conhecida como processamento da informação (*information process*).

Tikhomirov critica essa visão, baseado em pesquisas empíricas, em testes nos quais seres humanos e computadores deveriam resolver o mesmo problema. Ele argumenta que o computador trabalha os símbolos através de regras baseando-se na lógica matemática. Ou seja, trabalha os signos sem levar em consideração os seus significados, por isso mesmo não é possível atribuir juízo de valores a eles. Já os seres humanos operam no mundo real através dos símbolos com seus significados, em última análise, os humanos operam com os significados.

Outro argumento é o de que, no que diz respeito aos processos heurísticos envolvidos, ou seja, a como será resolvido o problema e até que problema deve-se resolver, a forma do computador resolver não se assemelha à forma de resolver do ser humano, exceto nas formas em que os problemas são de pouca complexidade. Processos heurísticos e valores afetam a maneira como o ser humano pensa. Portanto, se concebermos o pensamento do ser humano dessa forma não podemos aceitar que a relação da informática com o ser humano seja de substituição.

A teoria da complementação argumenta que o computador suplementa o pensamento, aumentando o volume e a velocidade de processamento humano. Tarefas seriam pensadas pelo ser humano e realizadas pela máquina informatizada, em separado, num processo de justaposição entre tecnologia e ser humano. Com o auxílio do computador, seres humanos podem processar mais informações, mais rapidamente, com mais precisão. Deste modo, não há influência da informática no pensamento.

Tikhomirov afirma que usar o computador não constitui uma adição puramente quantitativa de processos psicológicos já existentes. Se virmos a informatização em termos da ideia de mediação, os programas de computador devem ser vistos como um novo tipo de sistema de símbolos que podem mediar a atividade humana. Como Vygotsky e seus seguidores apontaram, quando uma nova forma de mediação é introduzida, a atividade não apenas amplia a capacidade da atividade existente mas, muitas vezes, também provoca uma melhoria qualitativa. O símbolo (linguagem, sinal matemático, mnemônico etc.) surge como um elo de mediação. No uso de meios auxiliares e sinais os seres humanos produzem mudanças em coisas externas, mas estas mudanças posteriormente provocam efeito sobre os seus processos mentais internos.

Como Tikhomirov argumentou, uma nova forma de mediação, como a fala, provoca uma etapa qualitativamente nova de pensar, uma nova forma de mediação, tal como a informatização, também dá origem a uma etapa qualitativamente nova. Dito de outra forma, a

informática não só suplementa como também modifica a forma de pensar do ser humano, estimulada pelas diversas realimentações que ocorrem quando usa a informática.

Por fim, Tikhomirov nos fala que, do ponto de vista educacional, devemos considerar esta nova entidade pensante chamada ser-humano-computador. Esta associação forte entre ser humano e tecnologia também é compartilhada por Pierre Lévy, assim como a crença de que o conhecimento é social e construído de forma coletiva.

Para Lévy, a história da humanidade sempre esteve permeada por tecnologias que expandem a nossa memória e colaboram para o conhecimento. São três as tecnologias da inteligência: oralidade, escrita e informática. Inicialmente a oralidade, que nos permitia recontar histórias preservando a cultura dos povos através de mitos. Depois a escrita, que teve significativo avanço após a invenção do livro. O texto escrito, além do ganho significativo para a memória da humanidade, também permitiu a revisão e a formalização de conceitos e a comunicação entre pessoas de diferentes épocas e lugares. A mídia escrita nos possibilitou organizar o mundo de forma linear, com princípio, meio e fim. A informática é a nova forma de extensão da memória da humanidade com diferenças qualitativas com relação às anteriores, sem abandoná-las. Ela permite a experimentação e a quebra da linearidade. Ela envolve a escrita, a oralidade, imagens, vídeos, tudo de forma dinâmica para além da comunicação instantânea. Esta nova tecnologia da inteligência nos desafia a outras formas de pensar, aprender e gerar conhecimento.

As ideias de Lévy têm forte relação com o advento da Internet e as relações que ele previu na sociedade. Sobre a Internet na sua forma de WWW, *World Wide Web*, disse Lévy, citado por Griebler, que é "um tapete de sentido tecido por milhões de pessoas e devolvido sempre ao tear. Da permanente costura pelas pontas de milhões de universos subjetivos emerge uma memória dinâmica, comum, 'objetivada', navegável" (Griebler, 2012, p. 3). A Internet representa uma inteligência coletiva, porque "é uma inteligência globalmente distribuída, incessantemente valorizada, coordenada em tempo real, que conduz a uma mobilização efetiva das competências" (Griebler, 2012, p. 3). Os participantes dessa inteligência coletiva podem compartilhar e colaborar para a formação de conhecimento de acordo com suas competências independentemente da sua localização no planeta.

Os espaços de aproximação na Internet são feitos pelos seus interesses e conhecimento, numa perspectiva inclusiva, uma vez que vivemos em espaços físicos, afetivos, sociais e históricos. Por exemplo, uma vizinha, cuja relação não vai além de cumprimentos no corredor,

está mais distante intelectualmente do que um autor morto há três séculos, que fica próximo através dos símbolos impressos em um livro. As pessoas que viajam no metrô estão mais afastadas no espaço afetivo, do que filha e pai que se encontram, pela Internet, a quinhentos quilômetros de distância.

As ideias de Tikhomirov de seres-humanos-computadores e as ideias de Lévy de inteligência coletiva levaram Marcelo Borba a cunhar a expressão seres-humanos-com-tecnologias, ou seres-humanos com mídias, numa perspectiva coletiva (Borba, 2002).

> A perspectiva histórica, a qual abraçamos, sugere que os seres humanos sejam constituídos por técnicas que estendem e modificam seu raciocínio e, ao mesmo tempo, esses mesmos seres humanos estão constantemente transformando essas técnicas. Assim, não faz sentido uma visão dicotômica. Mais ainda, entendemos que conhecimento só é produzido com uma determinada mídia, ou com uma tecnologia da inteligência. É por isso que adotamos uma perspectiva teórica que se apoia na noção de que o conhecimento é produzido por um coletivo formado por seres-humanos-com-mídias, ou seres-humanos-com-tecnologias e não, como sugere outras teorias, por seres humanos solitários ou coletivos formados apenas por seres humanos. (Borba, 2002, p. 3)

Para Borba, os coletivos de seres-humanos-com-tecnologias são a unidade epistemológica que devemos adotar para falar de conhecimento e conduzir as pesquisas sobre os processos de ensino-aprendizagem.

3.1.2.1 A teoria sócio-interacionista

A teoria sócio-interacionista foi inaugurada por Lev Vygotsky que destacou a importância da interação com os outros indivíduos e a força da cultura vigente na sociedade. Para Vygotsky o desenvolvimento psicológico do indivíduo se deve a fatores internos ao indivíduo – biológicos –, e externos – sociais e ambientais –, sendo, por isso, considerado um autor interacionista. Ele estabelece quatro Planos Genéticos de desenvolvimento: Filogênese, Ontogênese, Sociogênese e Microgênese (Oliveira, 1993; Rego, 2000). Filogênese – que é de natureza biológica e se refere à história de uma espécie animal (Natureza Biológica). Ontogênese – É também de natureza biológica e se refere ao desenvolvimento de um indivíduo de uma determinada espécie. Sociogênese ou Histórico Cultural – se refere às formas como a cultura interfere no desenvolvimento psicológico do indivíduo. Essa influência apresenta dois aspectos distintos. O primeiro é o alargamento das possibilidades humanas. Por exemplo, com a invenção do avião o homem passa a voar, ultrapassando as capacidades de natureza biológica. O segundo aspecto é como a cultura organiza e interpreta os eventos. Por exemplo, a adolescência, puberdade, é biológica, encarada de maneira

diferente por cada cultura. Microgênese – se refere aos fenômenos de aprendizagem por que passa aquele indivíduo. Tem um foco bem definido e limitado no tempo, entre o momento em que o indivíduo não sabe e o instante em que passa a saber algo. Como cada indivíduo tem uma história própria, ou seja, a sua própria microgênese, é através desse conceito que Vygotsky explica o não determinismo dos indivíduos. O determinismo de certa forma estava presente nos planos genéticos biológicos (Filogênese e Ontogênese) e culturais (Sociogênese).

Ainda segundo Vygotsky, o ser humano se relaciona com o mundo de forma mediada por ferramentas (faca, relógio, celular etc.). A mediação simbólica, através de signos, tem papel semelhante às ferramentas no campo psicológico. Os signos nos ajudam a lembrar, a passar uma ideia, a comparar, a escolher etc. O signo faz a conexão entre o sujeito e o objeto de conhecimento de forma semiótica (Oliveira, 1993).

A linguagem, por sua vez, usa signos. Signo é uma unidade linguística que contém um significante (forma ou imagem acústica) e um significado (conceito). Essa relação pode ser feita de modo arbitrário sem a necessidade de uma abstração direta. Os signos são construídos culturalmente, isto é, dependem de outras pessoas para o indivíduo aprender, manipular e resignificar. Vygotsky também chama a atenção para que, embora a palavra tenha um significado, o contexto pode atribuir-lhe vários sentidos.

> O sentido de uma palavra é a soma de todos os fatos psicológicos que ela desperta em nossa consciência. Assim, o sentido é sempre uma formação dinâmica, fluida, complexa, que tem várias zonas de estabilidade variada. O significado é apenas uma dessas zonas do sentido que a palavra adquire no contexto de algum discurso e, ademais, uma zona mais estável, uniforme e exata. Como se sabe, em contextos diferentes a palavra muda facilmente de sentido. (Barros, Paula, Pascual, Colaço, & Ximenes, 2009, p. 179)

Em outra referência a Vygotsky percebe-se que esta relação, entre sentido e significado, passa por conteúdos intelectuais e afetivos:

> Esse enriquecimento das palavras que o sentido lhes confere a partir do contexto é a lei fundamental da dinâmica do significado das palavras. A palavra incorpora, absorve de todo o contexto com que está entrelaçada os conteúdos intelectuais e afetivos e começa a significar mais e menos do que contém o seu significado quando a tomamos isoladamente e fora do contexto: mais, porque o círculo dos seus significados se amplia, adquirindo adicionalmente toda uma variedade de zonas preenchidas por um novo conteúdo; menos, porque o significado abstrato da palavra se limita e se restringe àquilo que ela significa apenas em um determinado contexto. (Barros et al., 2009, p. 179)

Sendo assim, cabe ao professor possibilitar ao aluno dar significado aos conteúdos programáticos exigidos socialmente através da escola. Para chegar ao significado usam-se

palavras, signos e sentidos para que o aluno signifique e ressignifique seus conceitos. A este propósito, Baktin esclarece que "não são palavras o que pronunciamos ou escutamos, mas verdades ou mentiras, coisas boas ou más, importantes ou triviais, agradáveis ou desagradáveis, etc. A palavra está sempre carregada de um conteúdo ou de um sentido ideológico ou vivencial" (Barros et al., 2009, p. 179).

É proferindo palavras ou outros signos carregados de sentidos, expostos em situações propícias, que vamos tentando estabelecer significados para novos signos e alterar os significados de outros já conhecidos, estabelecer os significados aceitos e desejados. Este seria o processo de aprendizagem. O processo através do qual se estabelece significação e ressignificação em situações didáticas que levam os alunos aos significados socialmente desejados passando por situações de desequilíbrio e reequilíbrio.

3.1.2.2 Passo cognitivo: entre o *Não Saber* e o *Saber*

A Zona de Desenvolvimento Proximal (ZDP) que Vygotsky definiu, é a distância entre o nível de competência de uma pessoa em realizar uma tarefa sozinho e o nível em que ela pode realizar a tarefa com auxílio de uma outra pessoa mais competente nesta tarefa. Durante este processo o aprendiz recebe ajuda adequada ao seu nível de conhecimento e interesse e aos poucos vai ganhando autonomia até conseguir fazer a atividade com independência. Esta ajuda vai sendo cada vez mais desnecessária à medida que ele se encaminha para a autonomia. Na Zona de Desenvolvimento Proximal os alunos modificam seus esquemas de conhecimento e seus significados e sentidos. Contudo, a ajuda adequada depende de cada pessoa e do momento, em função dos sentidos e significados que os alunos dão a cada situação. Por isso recomenda-se diversificar as abordagens (Coll et al., 2001). Ao referir-se à Zona de Desenvolvimento Proximal, Bruner diz que o mais apto empresta a consciência ao menos apto, dentro de uma transação negocial com o mundo, para que o menos apto adquira os símbolos e significados que fazem do mais apto ser reconhecido como mais apto (Correia, 2003).

David Shaffer e James Paul Gee observam que os videogames trabalham com desafios que não são nem tão difíceis de modo que causem frustração no jogador, nem tão fáceis a ponto de aborrecer o jogador. Este ponto ótimo é o estado de fruição e, assim como os jogos, as aulas deveriam se passar neste estado ótimo (Shaffer & Gee, 2012). Estes autores concordam com Resnick, quando afirmam que os alunos devem aprender a trabalhar com os outros e com ferramentas digitais para produzir conhecimento e não apenas para consumi-lo. Eles afirmam

que para o século XXI são necessárias habilidades como inovação, pensamento crítico e pensamento sistêmico. Shaffer e Gee afirmam que a tecnologia está mudando os comportamentos dos jovens, eles se organizam em comunidades de aprendizagem e chegam, como amadores, a atingir habilidades de profissionais em áreas como as histórias digitais. São fãs de ficção, de artes gráficas, de filmes caseiros, utilizando competentemente a computação gráfica, o desenvolvimento de jogos digitais, a fotografia digital, a robótica, entre outras habilidades.

Shaffer e Gee afirmam que devemos avaliar nossos alunos não apenas pelo conteúdo memorizado, mas também devemos incluir no mínimo a inovação, colaboração, produção e *design*. Esta lista mínima pode ser acrescida de participação cidadã, pensamento crítico, pensamento sistêmico, habilidades técnicas, capacidade de produzir com a mídia digital etc. (Shaffer & Gee, 2012).

Seymour Papert (Papert, 2002) fala de um novo tipo de divertimento. O "Hard Fun", algo como difícil mas divertido. Ele fala que a sua intenção é levar aos alunos uma forma mais engajada e menos coercitiva de aprendizagem, conduzindo à autodisciplina para atingir objetivos. Ele afirma que as pessoas estão dispostas a enfrentar desafios, desde que estejam alinhados com seus interesses e sua cultura. Isso leva o educador a trabalhar no sentido de tornar o trabalho do aprendiz difícil da forma correta, conectando a aprendizagem com os interesses das crianças, com as áreas de conhecimento e com o perfil ético necessário ao mundo dos adultos. Em outras palavras, Papert nos diz que se pode trabalhar na zona de fluidez com altos níveis de dificuldade (*hard*) desde que o aluno se sinta recompensado (*fun*). Por outro lado, Bruner afirma que o desenvolvimento cognitivo será tanto mais rápido quanto mais rico e estimulante for o meio cultural (Correia, 2003), isto é, para conseguir a motivação do aluno, o professor tem que observar o que o estimula, e a cultura que o cerca é uma grande aliada nesta busca.

Estas perspectivas sobre o desenvolvimento cognitivo leva o professor a ter que considerar o aluno em outros contextos culturais, além da sala de aula. Veja Figura 2.

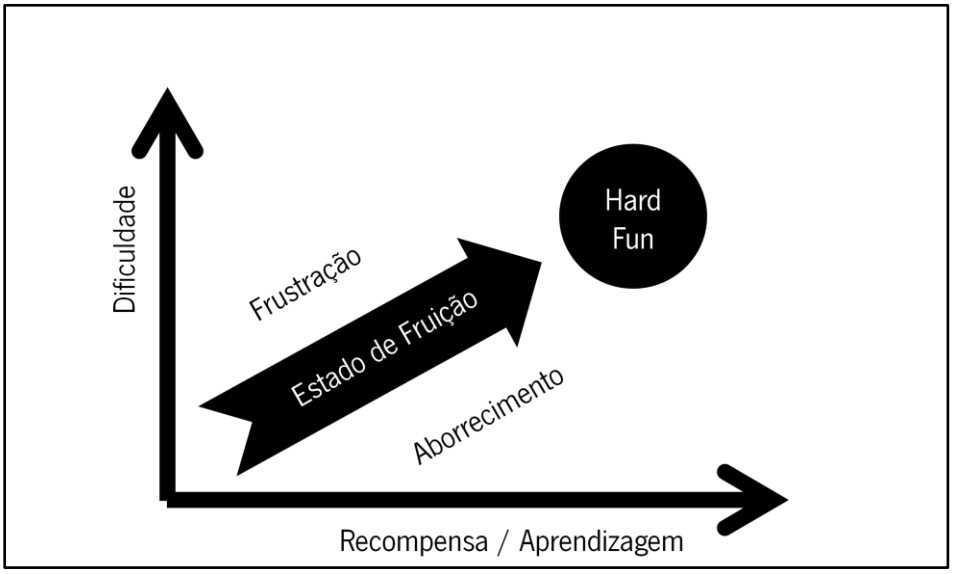


Figura 2- Região de Fluidez

3.1.3 Aprendizagem na era digital

Mitchel Resnick (2002) constata que os computadores costumam ser utilizados na educação para manter antigas abordagens. Segundo ele, a concepção de computadores e educação centrada na informação limita e distorce os campos da educação e da computação.

Indo além da concepção de que ensinar é depositar conhecimentos Resnick escreve:

> Os professores não podem simplesmente despejar informações na cabeça dos alunos; em vez disso, a aprendizagem é um processo ativo no qual as pessoas constroem novas compreensões do mundo ao seu redor através da exploração ativa, experimentação, discussão e reflexão. Em suma: as pessoas não adquirem ideias; pessoas constroem ideias. (Resnick, 2002, p. 2)

Computadores são ótimos para armazenar e divulgar informações, mas também são ferramentas de criação e expressão das pessoas. As pessoas aprendem mais quando estão engajadas em projetar e criar coisas, especialmente quando estas coisas têm significado para elas. Por isso, Resnick propõe que:

> Assim como a pintura a dedo (e ao contrário da televisão), os computadores podem ser usados para projetar e criar coisas. Além de acessar páginas da Web, as pessoas podem criar suas próprias páginas web. Além de baixar os arquivos de música MP3, as pessoas podem compor suas próprias músicas. Além de jogar SimCity, as pessoas podem criar seus próprios mundos simulados... Na verdade, o computador é o material de construção mais extraordinário já inventado, permitindo às pessoas construírem qualquer coisa desde vídeos de música até simulações científicas para criaturas robóticas. Os computadores podem ser vistos como um material de construção

> universal, expandindo o que as pessoas podem criar e o que eles podem aprender no processo. (Resnick, 2002, p. 2)

Outra questão abordada por Resnick é a Fluência Digital. Quando as pessoas são introduzidas a usar computadores, são conduzidas a fazer tarefas mecânicas, como enviar um e-mail ou usar o editor de texto. Para ser fluente a pessoa tem que não apenas usar o computador, mas também ser capaz de utilizá-lo para construir coisas que tenham significado. A fluência digital se tornará um pré-requisito para se obter um emprego e participar significativamente na sociedade e tem efeito catalisador para o aprendizado durante toda a vida. A tendência é que a falta de oportunidade para os que não têm acesso ao computador vá diminuindo à medida que o preço dos computadores cai, restando a questão da falta de fluência digital como limitadora das oportunidades sociais.

Resnick critica a ideia estabelecida de salas de aula. O modelo vigente é o de um professor que distribui conhecimento a uma sala repleta de alunos, aproximadamente da mesma idade, em diferentes disciplinas separadas por períodos de tempo de uma hora. Este modelo caracteriza uma forma passiva de aprendizado, ao que Resnick contrapõe uma forma mais ativa e empreendedora, na qual alunos e professores atuem em projetos interdisciplinares, tendo o professor como consultor, em períodos longos de tempo. Nestes projetos seria possível ir mais a fundo e com mais significado em ideias que surgem durante a evolução do trabalho.

Resnick reconhece que é necessário adequar os computadores e o currículo para as novas gerações, dado que muitos dos programas existentes são feitos para adultos utilizarem no ambiente de trabalho. Enquanto isso, os jovens de hoje estão ansiosos por fazer mais com o computador e necessitam de ferramentas adequadas a essa nova geração. Para Resnick, muito do que hoje é ensinado na escola vem da era do lápis e do papel. Faz-se necessária uma mudança nos currículos escolares para preparar os estudantes para as novas habilidades e ideias caracterizadoras da sociedade digital.

Outra razão para a mudança no currículo é que a tecnologia está mudando, não só o que se deve aprender, mas também como e o que se pode aprender. Por exemplo, o estudante pode aprender com simuladores em seu computador. Simulações de sistemas econômicos, ecossistemas e sistemas imunológicos, o que seria muito mais difícil de aprender com lápis, papel, livros e quadro-negro. O mais importante é transformar o currículo de modo a se obter menos conteúdo a ser aprendido e mais estratégias para se aprender. Quando se pensa em reforma do sistema educativo, pensa-se que o aprendizado é feito apenas nas escolas, nos

horários das aulas e entre as idades de 6 a 18. Mas, segundo Resnick, as iniciativas nacionais de educação devem considerar que, na era digital, se aprende em todos os lugares, o dia inteiro e em todos os dias da vida. "Iniciativas nacionais de educação devem ter como objetivo melhorar as oportunidades de aprendizagem, não só nas escolas, mas também em casas, centros comunitários, museus e locais de trabalho" (Resnick, 2002, p. 5).

3.1.3.1 Aprendizagem no estilo Jardim de Infância

O Jardim de Infância é um conceito criado por Friedrich Fröebel e concretizado no ano de 1840 em Bad Blankenburg, na Alemanha. O nome Jardim de Infância (Kindergarten) é uma metáfora que alude à necessidade de proteção e cuidados que crianças tão pequeninas precisam. Nesta escola, a professora seria a jardineira da educação infantil, proporcionando o desenvolvimento intelectual, emocional, físico, social e moral das crianças pelo uso de jogos, atividades semelhantes às do jardineiro que cuida das suas plantas. Estas escolas eram destinadas a crianças de 3 a 7 anos (Arce, 2004).

As teorias de Fröebel se baseiam em crenças religiosas e metafísicas, junto com a visão romântica de época segundo a qual, na sua essência, o homem é bom e deve se integrar na natureza. A escolha da metáfora do Jardim de Infância e da professora como jardineira já retrata a visão de Fröebel, coerente com sua época. Fröebel hesitou em chamar-lhe de escola evitando a conotação com formatar mentes, pois este espaço deveria servir para cultivar nas crianças suas tendências divinas através de ocupações e atividades livres (Kishimoto, 1996).

Fröebel defende que a vocação do homem é para ser livre, produtivo e criativo, sendo a liberdade um elemento essencial. Em razão disso, a educação, inicialmente prescritiva, categórica e interferidora deve, aos poucos, dar espaço para a livre e espontânea representação do divino no homem. Ao considerar que o homem é a imagem e semelhante a Deus, Fröebel atribui à criança as qualidades de ser produtora e criativa (Kishimoto, 1996).

Outros já ressaltaram a importância dos jogos na educação. Platão, em *As Leis*, destaca a importância do aprender brincando em oposição à violência e a repressão. No Renascimento a brincadeira já é vista como uma ação livre que estimula o desenvolvimento da inteligência e facilita o estudo, contrapondo-se aos recursos de verbalização e palmatória, que se usavam na época, porém Fröebel é o primeiro a colocar a brincadeira como parte essencial do trabalho pedagógico (Kishimoto, 1996).

Aos brinquedos criados para auxiliar no processo educativo, Fröebel dava o nome de dons e estes seriam presentes dados às crianças para auxiliá-las a descobrir seus próprios dons que, por sua vez, seriam os presentes de Deus para cada uma delas (Arce, 2004). Objetos como bolas, cubos, varetas, anéis, material de desenho, régua, fitas, dobraduras etc., materiais utilizados nas ocupações conduzidas pela jardineira, são dons para Fröebel.

No jardim, para além das ocupações havia os momentos dos jogos, momentos em que os dons eram utilizados livremente e as brincadeiras, atividades simbólicas e livres, eram acompanhadas de músicas e movimentos corporais, destinadas a liberar a criança para expressar as relações sobre objetos e situações do dia-a-dia. As ocupações podem ser entendidas como instrumentos da educação enquanto as brincadeiras livres eram propostas para a criança exercer o poder de conquistar a natureza. Os jogos, por sua vez, tinham uma função em si mesmos de autoexpressão e espontaneidade e outra função de busca de resultados.

Susan Blow, já em 1911, aponta alguns equívocos das jardineiras ao aplicar estes métodos, descobrindo falhas na compreensão da atividade criativa e comprometendo a busca da harmonia entre direção e liberdade, entre individual e coletivo. As atividades livres são comprometidas em função das ocupações. Os dons e as ocupações passam a ser recursos didáticos para ensinar formas estéticas e conteúdos, ignorando as imitações espontâneas e os jogos de manipulação livre (Kishimoto, 1996).

Embora haja críticas às bases metafísicas e de cunho religioso na teoria do Jardim de Infância, as práticas são corroboradas por autores mais atuais da escola russa, como Alexei Leontiev, Daniil Elkonin e Lev Vygotsky.

> Ao utilizar um complexo sistema metafísico de relações entre a natureza divina e as conexões internas do ser humano, edificado pelo princípio de unidade vital e totalidade, para explicitar o desenvolvimento do simbolismo infantil, sua teoria apresenta difícil sustentação. Sua psicologia assenta-se nessa filosofia metafísica e não na descrição de processos psicológicos infantis. Entretanto, suas práticas pedagógicas intuitivas, relacionadas às brincadeiras interativas e motoras mãe-criança, fruto de observações sistemáticas de brincadeiras maternas, não se distanciam dos pesquisadores que estudam essa questão nos tempos atuais. (Kishimoto, 1996, p. 20)

Elkonin, Leontiev e Vygotsky rejeitam a hipótese do ser humano divino e espiritual como base para a ciência. Eles partem das bases Marxistas, nas quais o homem define e é definido pela sua própria história. Para a escola russa, as condições culturais, econômicas, sociais e históricas são fatores decisivos no desenvolvimento infantil. Sendo assim, a educação não pode

ser única, divina, natural e universal como propõe Fröebel. Na visão russa, a passagem de uma fase para outra não se deve a uma simples evolução, mas sim a uma revolução resultante de um processo dialético de interações sociais que geram necessidades e motivos para o seu desenvolvimento intelectual com ganhos qualitativos.

A brincadeira também é vista como atividade principal para os pedagogos russos, mas de um ponto de vista diferente. As brincadeiras de faz-de-conta não são uma forma de se distanciar da realidade, mas sim de possibilitar que as crianças se aproximem do mundo dos adultos usando a fantasia como auxílio, superando assim suas impossibilidades. Ao interpretar um motorista de ônibus, a criança substitui as operações necessárias por outras que estão mais dentro de suas possibilidades. Sendo assim, a criança não está se distanciando da realidade e sim se aproximando dela. Estas atividades são intencionais e não instintivas, como supõe Fröebel. Leontiev afirma que a quebra da relação entre significado e sentido, estabelecida durante a brincadeira, é abandonada assim que a criança deixa de brincar. Isso quer dizer que a criança tem consciência da sua fantasia e que ela age no seu cotidiano motivada pela realidade objetiva.

De acordo com a corrente russa, a brincadeira é a atividade principal da criança porque provoca os mais importantes desenvolvimentos psíquicos com ganhos qualitativos. A brincadeira cria zonas de desenvolvimento proximal possibilitando que a criança realize ações além das que a sua idade lhe permite realizar. Quando a criança usa um cabo de vassoura para andar a cavalo é porque ela tem como alvo o processo social e não a ação em si.

Mitchel Resnick também defende que o estilo de aprendizagem adotado pelo jardim de infância é adequado às necessidades para a sociedade do século XXI. Resnick (2007) observa um ciclo no processo de aprendizagem que define como sendo *Imaginar, Criar, Brincar, Compartilhar, Refletir* e novamente *Imaginar*. Este investigador advoga que artefatos tecnológicos sejam criados para entender esta experiência de aprendizagem de jardim da infância em outras fases do aprendizado a ser realizado durante toda a vida.

Imaginar – Fröebel propõe para o jardim da infância uma lista de materiais didáticos a que chama de dons, dados às crianças para que elas descubram os seus talentos, seus dotes naturais ou seus verdadeiros dons. Na prática eles servem para estimular a imaginação sem conduzir ou limitar em excesso. Com estes materiais os alunos conseguem brincar, experimentar e aprender os conceitos básicos do jardim de infância. Resnick propõe que se faça, com base em tecnologias atuais, materiais de experimentação, levando esta experiência de

jardim da infância para as demais faixas etárias, possibilitando um aprendizado no estilo jardim de infância para conceitos mais avançados e por toda a vida. Como exemplo, ele cita dois artefatos tecnológicos desenvolvidos no Laboratório de Mídia do Instituto de Tecnologia de Massachusetts que são o ambiente de programação Scratch e os Crickets.

Estes novos artefatos tecnológicos não devem conduzir em demasia o aluno em um processo de ensino. Eles devem ser suficientemente versáteis de modo a respeitar os diversos estilos de aprendizagem dos alunos.

Criar – Resnick recorda que Fröebel foi o precursor do construcionismo e que seus dons possibilitavam que os alunos usassem a sua imaginação para se engajarem significativamente em criar seus projetos pessoais e critica os brinquedos eletrônicos disponíveis no mercado, porque, na sua maioria, não possibilitam a criação, apenas a interação.

Brincar – Piaget dizia que brincar é o trabalho da criança (Resnick, 2007, p. 3). Embora brincar seja bem aceito para as crianças do jardim de infância, educadores e pais tendem a separar aprender e brincar, de modo que, brincar passa a ser uma atitude de pouca valia para a aprendizagem e até mesmo antagônica a ela. Para Resnick, brincar e aprender podem e devem estar interligados. Ambos envolvem experimentação, exploração e teste de limites.

Mesmo algumas relações entre educação e brincar são estabelecidas em um estilo que contraria o estilo de aprendizagem do jardim da infância. Neste contexto vale apena mencionar as recentes iniciativas de Entretenimento Educativo (*Edutainment*). A este respeito, Resnick critica a visão de que a educação é amarga e de que deve ser misturada com o entretenimento para que seja palatável. Esta concepção tem como base a ideia de que entretenimento é divertido e educação não é, que educação é necessariamente a ausência de diversão. Outra crítica que faz dirige-se ao aspecto de que tanto a educação como o entretenimento estão relacionados a empresas que fornecem estes serviços feitos para que o aluno consuma de forma passiva. Esta passividade não está de acordo com a intenção de formar pessoas criativas e pensantes (Resnick, 2007).

Com a extraordinária popularidade dos jogos eletrônicos na sociedade atual, muitos pesquisadores passam a estudar como e o que se aprende com os *videogames*. Porém, mesmo os *videogames,* que produzem um extraordinário engajamento por parte dos alunos, não permitem um aprendizado no estilo jardim de infância, na medida em que não proporcionam espaço para projetar e criar. Resnick admite como exceção os jogos de simulações e os Jogos Epistêmicos de Shaffer. Yasmin Kafai (Resnick, 2007, p. 4), em suas pesquisas, mostra como

crianças da escola primária se tornam pensadores criativos ao criarem seus próprios *videogames*. Sendo assim, o Scratch pode ser uma alternativa para que as crianças desenvolvam seus próprios jogos, experimentando e testando limites, ao estilo do jardim de infância.

Compartilhar – Assim como a brincadeira, compartilhar é valorizado no jardim de infância, e da mesma forma, também é deixado para trás nos anos escolares seguintes. Porém, Resnick aponta três fatores que estão revertendo esta tendência:

- Empresários perceberam que o trabalho colaborativo é mais importante no ambiente de trabalho hoje em dia do que jamais foi. Isto levou-os a encorajar as escolas a incentivar a colaboração para o mundo do trabalho.
- A tendência educacional sócio-histórica, representada por Vygotsky, que tende a promover o aprendizado natural através da promoção de comunidades de aprendizes.
- A proliferação do acesso à Internet e de tecnologias interativas, conduzindo a uma cultura participativa e criando e produzindo conteúdo na web através de blogues, *sites* colaborativos e redes sociais.

Tecnicamente, o ambiente Scratch facilita a curva de aprendizagem da linguagem e oferece um *site* onde os produtos podem ser compartilhados. O *site* conta com uma extensa variedade de programas e também promove a disseminação da linguagem de programação, servindo de consulta aos demais programadores. Esta solução proporciona inspiração e uma vitrine para mostrar o trabalho criado, alterado ou iniciado por um programador e compartilhado com os outros.

Refletir – Resnick cita a escola de Reggio Emília para falar da documentação dos projetos realizados pelas crianças. A escola Reggio Emília foi fundada após a segunda guerra mundial com a participação intensa da comunidade, na pequena cidade de Reggio Emilia no norte da Itália. Ela foi construída com a venda de um tanque de guerra, seis cavalos e três caminhões, deixados pelos alemães.

> Esse movimento inicial envolveu toda a comunidade, mas de modo especial os pais, pois nasceu do desejo de reconstrução da própria história e da possibilidade de uma vida melhor para seus filhos. Então, desde sua origem, Reggio Emilia é uma escola diferente, enraizada na vontade das famílias de construir um mundo melhor por meio da educação. (Sá, 2010, p. 3)

Em abril de 1945 Loris Malaguzzi se junta aos camponeses e se torna o mentor de uma escola revolucionária cujo processo de ensino-aprendizagem se expandiu para outras escolas nos subúrbios e nos bairros mais pobres da cidade. As escolas de Reggio Emilia são reverenciadas até aos dias de hoje, por adotarem um ensino não convencional, centrado no aluno e no sucesso (Sá, 2010).

A escola Reggio Emilia tem o costume de colocar a documentação do que foi produzido pelos alunos nas paredes e em pastas acessíveis a todos (Sá, 2010). Resnick realça que as paredes ficam repletas de desenhos e esquemas dos alunos e anotações dos professores. Os professores usam estes artefatos para fazer com que os alunos reflitam sobre o trabalho realizado. Resnick conclui que não adianta focar na realização de projetos, na produção de artefatos sem a devida reflexão. A reflexão crítica desses artefatos levará o aluno a entender as ideias que guiaram o projeto, suas melhorias e as relações entre concepções científicas e sua relação com os fenômenos do mundo real.

Nas oficinas realizadas no MIT Media Lab também é explicitamente exposto o ciclo de desenvolvimento Imaginar-Criar-Brincar-Compartilhar-Refletir. Propondo esta reflexão sobre o próprio trabalho e a explicitação do processo, Resnick ensina a resolução de problemas, indo ao encontro das ideias de Pea e Kurland (1984), quando estes afirmam que, por si só, aprender a programar não garante o aprendizado de resolução de problemas. Estes conteúdos têm que ser especificamente abordados como propõe Resnick.

3.1.3.2 A importância da criatividade na Educação

Nos anos de 1980 era tema de conversa a mudança que se fazia presente da Sociedade Industrial para a Sociedade da Informação (Resnick, 2002), uma vez que os recursos naturais e os manufaturados já não determinavam tão fortemente a economia e a sociedade. Na década seguinte se falava de uma Sociedade do Conhecimento, na qual a informação, simplesmente, não traria mudanças importantes, mas a transformação de informação em conhecimento e a gestão desse conhecimento passaram a ser importantes. A sociedade do futuro será supostamente a Sociedade Criativa.

> A meu ver, o sucesso no futuro será baseado não em quanto nós sabemos, mas na nossa capacidade de pensar e agir criativamente. A proliferação das tecnologias digitais tem acentuado a necessidade de pensamento criativo em todos os aspectos de nossas vidas, e também forneceu ferramentas que podem ajudar-nos a melhorar e reinventar a nós mesmos. Em todo o mundo, as tecnologias de computação e comunicações, está

> gerando um novo espírito empreendedor, a criação de produtos e serviços inovadores, e aumento da produtividade. A importância de um cidadão criativo e com uma boa educação é maior do que nunca. As crianças devem ter um papel central nesta transição para a Sociedade Criativa. A infância é um dos períodos mais criativos da nossa vida. Temos de nos certificar de que a criatividade das crianças é alimentada e desenvolvida, e devemos ajudar as crianças a aprender a ampliar e refinar suas habilidades criativas, a fim de que a criatividade da infância persista e cresça ao longo da vida. Para atingir esses objetivos, serão necessárias novas abordagens para o ensino e aprendizagem, e novos tipos de tecnologias para apoiar essas novas abordagens. O objetivo final é uma sociedade de indivíduos criativos que estão constantemente inventando novas possibilidades para si e para suas comunidades. (Resnick, 2002, p. 6)

Alexander Luria, ao contrário de Resnick, afirma que não se pode dizer que a criança é mais criativa que o adulto, porém o pensamento da criança é mais fantasioso enquanto o do adulto é mais dirigido à realidade (Moysés, 1997). Contudo, Luria afirma que a criatividade pode ser desenvolvida e, nesse ponto, está de acordo com Resnick. Outro aspecto a destacar na criatividade, referido por Resnick, é o seu caráter ligado à utilidade, indo além da simples originalidade. Este aspecto da utilidade relacionado a criatividade também é ressaltado por Maria de Fátima Morais (Morais, 2012).

No entanto, na minha experiência, eu vejo que os universitários recém- ingressados, aos quais ministro minhas aulas, sentem dificuldade em pensar criativamente. No entanto, assim como Maltempi & Valente, eu acredito que, de entre os selecionados na cultura escolar que temos hoje, que tolhe a criatividade, eles são os mais bem adaptados.

> Atualmente, a maioria dos alunos que chegam às universidades apresenta deficiência relacionada com raciocínio lógico e criatividade. Isso porque eles são provenientes de um sistema educacional que privilegia e exige a memorização e reprodução de informações, além de punir o erro. Tais características praticamente inviabilizam o pensamento crítico e criativo. (Maltempi & Valente, 2000, p. 1)

Em seu artigo "Criatividade: investimento pessoal e organizacional para o séc. XXI?", Fátima Morais (2012) nos fala das múltiplas definições de criatividade e da dificuldade de se abarcar o conceito em sua totalidade. Porém, ela destaca o caráter pragmático da criatividade, indo além da originalidade e de sua utilidade no cotidiano, sem a pretensão de nos compararmos a um Mozart ou Saramago. A investigadora realça que, independentemente da sua distribuição entre a população, a criatividade é algo que todos nós possuímos. Mas a criatividade é passível de desenvolvimento, relacionando-se com a riqueza das experiências e conhecimentos prévios (Moysés, 1997).

Morais lista alguns requisitos encontrados em pessoas criativas, mas alerta que a existência dos requisitos não garante a criatividade. Os requisitos devem ser entendidos como recomendações para o que se deve incentivar nas práticas de sala de aula visando a criatividade (ver Tabela 3).

Tabela 3 – Requisitos da Criatividade (Morais, 2012)

Requisitos	Comentário
Aptidões	As pessoas têm uma certa inclinação para uma determinada área como por exemplo aptidão verbal ou figurativa.
Conhecimento	Conhecimento específico, por exemplo, ninguém pode ser criativo em música se não souber música. É preciso um conhecimento multidisciplinar para fazer associações de informações.
Motivação	A motivação pode ser intrínseca ou extrínseca, mas a pessoa deve estar comprometida emocionalmente.
Personalidade	Características da personalidade: Autonomia, Autoconfiança, Abertura à experiência, Curiosidade, Sentido de humor, Tolerância à ambiguidade, Tomada de risco, Sensibilidade estética, Paixão pelo que se faz, Atração pela complexidade, Persistência.
Processos	São processos cognitivos criativos: flexibilidade de perceptiva, uso de imagens, pensamento analógico e metafórico, criação e da descoberta de problemas (não apenas a sua resolução).
Olhar do Outro	Criatividade é uma atribuição dada por outro. Reflete o momento socio-histórico com seus valores, interesses, atitudes e conhecimentos.

3.1.3.3 Brinquedos do século XXI e aprendizagem

Alan Kay conta que nas populações tradicionais as crianças podem aprender por imitação dos adultos, uma vez que o que se precisa para ser um cidadão integrado na cultura se mostra claramente no dia a dia observado pela criança. No entanto, muitas das habilidades necessárias à sociedade moderna não se mostram claramente na vida das nossas crianças. Neste caso devemos proporcionar a estas crianças brinquedos que exercitem as ideias compatíveis com nossa era (Kay, 2003).

A ciência está sendo ensinada nas escolas de forma dogmática, sem a possibilidade de experimentação, mas, segundo Kay deveria ser ensinada como se ensina um construtor de mapas, como sendo uma versão do mundo, embora nunca uma versão definitiva (Kay, 2007). Kay menciona duas contribuições para um mundo de simulações. A primeira é a ideia de que múltiplos objetos coexistem independentemente e podem colaborar entre si. A segunda é que os ambientes de programação devem ser adequadamente projetados com especial atenção para a interface do usuário (Kay, 2007). Detalhes, como a regra de “7 mais ou menos 2” ou Lei de

Miller (1956), devem ser observados. Esta regra se refere à natural capacidade da mente humana em se concentrar em 5 a 9 itens ao mesmo tempo, e não em mais do que isso. Princípios como este devem ser levados em consideração para que o usuário iniciante não se sinta incapaz de aprender. Outro detalhe é, por exemplo, o duplo sentido que tem o sinal de igual (=), ora representando uma atribuição de valor a uma variável, ora representando uma comparação entre quantidades. Isso pode causar confusão para as crianças. Por fim, Kay também se mostra crítico em relação ao uso de *software* fechado que não dá margem para modificação do modelo com relação à realidade. Por exemplo, o SimCity leva a crer que a solução para o aumento da criminalidade seria o aumento do número de delegacias, solução de que antropólogos, sociólogos, psicólogos e economistas descordam seriamente. Isso acontece porque os pacotes de *software* utilizam regras arbitrárias, que não temos como questionar ou alterar (Kay, 2007).

Alan Kay alega que devemos criar para as crianças brinquedos compatíveis com nossos tempos e Resnick acrescenta que devemos criar brinquedos que estimulem a aprendizagem ao estilo jardim de infância por toda a vida. Crickets (PicoCrickets), Scratch e Geogebra são exemplos desse tipo de brinquedos.

Crickets – eram pequenos dispositivos programáveis, a ponto de caber na palma da mão de uma criança. Com os Crickets era possível ligar motores, luzes e outros dispositivos; era possível ler sensores de áudio, luz e toque. Os Crickets possibilitavam que as crianças criassem seus próprios brinquedos, exercitando sua imaginação. Um grupo de meninas em uma escola de Boston usou os Crickets e outros materiais artesanais para criar um jardim interativo, com flores que dançavam e mudavam de cor quando se batia palmas. Um menino de 12 anos em Hong Kong, criou um *jukebox* que tocava músicas diferentes, dependendo do tipo de moeda que se colocava. Uma menina de 11 anos colocou luzes nas suas botas, acendendo diferentes cores dependendo do ritmo de suas passadas (Resnick, 2007). Após seu um período de prosperidade entre 1996 e 2010 os Crickets foram descontinuados.

Scratch foi desenvolvido para os iniciantes em programação com idade a partir de 8 anos. O ambiente de programação tem base nas ferramentas Logo e Etoys, com a intenção de promover o construcionismo. O Scratch é gratuito, está traduzido para mais de 50 línguas e tem mais de dois milhões de cópias baixadas. Os comandos estão representados em blocos que se encaixam, inspirados na montagem dos blocos do brinquedo Lego. Este recurso evita erros de sintaxe. Existe um número reduzido de comandos, porém versáteis o suficiente para realizar

uma ampla gama de algoritmos. Pode-se executar um trecho do programa com um clique duplo diretamente no comando ou bloco de comandos, facilitando a depuração. Ao executar é mostrado em tempo real o valor das variáveis e é destacado com um contorno branco o bloco que está sendo executado, o que auxilia no entendimento do programa descrito e respectiva execução. Quando acontece um erro em um bloco (por exemplo, uma divisão por zero), o bloco fica com contorno vermelho em vez de branco. Também é possível executar dois blocos de comando em paralelo com facilidade, disponibilizando um recurso poderoso aos programadores (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010; Resnick et al., 2009).

O projeto do Scratch tem a intenção de simplificar para os que ainda não praticam a programação de computadores. Sendo assim, em algumas decisões entre simplicidade e versatilidade optou-se pela simplicidade. Por este motivo, o Scratch não é uma linguagem orientada a objetos, apesar de ser baseada em objetos. No Scratch existem objetos que armazenam estados (variáveis, imagens, sons) e comportamentos (comados), mas não existem classes ou herança, elementos característicos das linguagens orientadas a objeto. No entanto, a forma de pensar, baseado na metáfora ou modelo de objetos, se mantém. Outra simplificação vem do fato de apenas os comandos do objeto serem capazes de mudar o status do próprio objeto. Por exemplo, apenas o objeto Gato pode executar o som Miau que está armazenado no objeto Gato. No entanto a interação é permitida com mensagens enviadas a todos os objetos (*broadcast*). Com o Scratch são criados jogos, histórias animadas, adaptações de livros, mensagens de felicitações, vídeos de música, projetos de ciência, tutoriais, projetos de arte e música. O projeto Scratch se mostra bem sucedido na tentativa de exigir pouco para iniciar a programar e possibilitar muito em possibilidades de programação. Como afirmam Maloney e seus colegas:

> O ambiente de programação Scratch e sua linguagem trabalham em conjunto para criar um sistema que é excepcionalmente rápido, possibilitando que os usuários aprendizes comecem a programar dentro de quinze minutos, mas com suficiente profundidade e variedade para mantê-los engajados por anos. (Maloney et al., 2010, p. 14)

O Geogebra é um software de geometria dinâmica desenvolvido e mantido por Markus Hohenwater desde 2001. Este software integra as representações geométricas, algébricas e numéricas e é destinado aos jovens de 10 a 18 anos e a professores do ensino médio. Com ele é possível, por exemplo, manipular um círculo diretamente e ver o que acontece com a equação do círculo, ou manipular a equação e ver como fica a representação na janela de geometria (Hohenwarter & Fuchs, 2004). Geogebra oferece pontos, vetores, ângulos, segmentos,

polígonos, linhas retas, todas as seções cônicas e funções em x, como se fossem os dons de Fröebel. A partir desses elementos (dons) os alunos podem experimentar e conjecturar a respeito desses objetos matemáticos. Essas conjecturas são o verdadeiro "fazer matemática". A experimentação intensa com estes entes abstratos muda o jeito como se aprende a matemática, valendo-se de múltiplas representações que, por sua vez, nos dão várias visões do mesmo objeto matemático complementando nosso conceito do que seria este objeto.

3.1.4 Aprendizagem com mínima intervenção

Os computadores, como os bons aplicativos de interfaces intuitivas, fazem com que muitas das habilidades necessárias para a manipulação de computadores sejam aprendidas sem um professor. Mas até onde pode ir este aprendizado com o mínimo de intervenção? Quais os processos e estratégias que se formam a partir do coletivo para o aprendizado do coletivo?

Sugata Mitra (2000) nos conta a experiência realizada com quiosques "Hole in the Wall", na qual o computador fica atrás de uma parede e crianças de comunidades isoladas têm livre acesso e uso sem pedir permissão. As comunidades isoladas ou distantes podem ser comunidades fisicamente distantes (até 300km de Nova Deli) ou comunidades economicamente distantes como os bolsões de pobreza que vemos em comunidades urbanas. Mitra relata que estas populações de crianças desfavorecidas se alfabetizam sozinhas em informática por um processo que ele descreve em oito tópicos:

> 1. Uma criança explora de forma aleatória o ambiente GUI (Graphical User Interface), outros assistem até que uma descoberta acidental é feita. Por exemplo, quando descobrem que o cursor se transforma em uma forma de mão em certos lugares na tela.
> 2. Várias crianças que repitam a descoberta, solicitando à primeira criança que o deixe repetir.
> 3. Quando a criança está descobrindo ou repetindo a etapa 2, ela realiza mais descobertas ocasionais.
> 4. Todas as crianças repetem todas as descobertas feitas e, no processo, faz mais descobertas e começam a criar um vocabulário para descrever a sua experiência.
> 5. O vocabulário as incentiva a perceber generalizações (exemplo: quando você clica em um cursor em forma de mão, ele muda para uma ampulheta por alguns instantes e depois aparece uma nova página)
> 6. As crianças memorizaram procedimentos inteiros para fazer alguma coisa, por exemplo, como abrir um programa de pintura e recuperar uma imagem salva. Eles ensinam uns aos outros caminhos mais curtos para fazer a mesma coisa, sempre que um deles encontra um novo procedimento mais eficiente.
> 7. O grupo se divide em os que "sabem" e o que "não sabem", assim como eles fizeram entre os que "possuem" e os que "não possuem". No entanto, eles percebem

> que uma criança que sabe que irá partilhar seu saber em troca de amizade, em oposição ao possuir das coisas físicas, onde eles poderiam usar a força para conseguir.
> 8. Um estágio é alcançado quando novas descobertas são feitas e as crianças se ocupam em praticar o que já aprenderam. Neste ponto é necessária uma intervenção de um adulto para introduzir uma nova "semente" de descoberta ("você sabia que computadores podem tocar música? Aqui, me deixe tocar uma música para você"). Normalmente, uma espiral de descobertas se segue e outro ciclo de autoinstrução começa. (Mitra, 2000, p. 17)

Sugata Mitra (2000) finaliza dizendo acreditar que assim como este modelo de educação com mínima invasão se mostrou promissor para o ensino e um letramento em informática, uma pedagogia semelhante deve servir para outros conteúdos a serem ensinados.

Parimala Inamdar e Arun Kulkarni (2007) realizaram um experimento para saber se o método educacional de mínima invasão pode impactar nos resultados escolares. Nesse experimento elegeu-se matemática, inglês e ciências como os conteúdos a serem estudados. Mostrou-se o fortalecimento das competências em matemática, mas não houve um impacto significativo em ciências e inglês. Em conversas com o professor de matemática Shamsuddin Attar, na Shirgao School, sugeriu-se que os exames escolares de inglês e ciência na Índia são amplamente baseados em responder a perguntas de livros didáticos de uma forma prescrita, fortemente baseada em memorização, caracterizando uma aprendizagem mecânica sem significado para os alunos. Enquanto isso, o exame de matemática da escola, por natureza, tem a ver com a resolução de problemas. Na opinião de Attar, estas capacidades de resolução de problemas básicos desenvolvidas através da aprendizagem autodirigida em grupo, através do quiosque de computador, poderiam impactar na pontuação das provas escolares na disciplina de matemática

Sugata Mitra e Ritu Dangwal (Mitra & Dangwal, 2010) realizaram outros experimentos para identificar os limites do autoaprendizado em grupos de crianças do projeto "Hole in the Wall", focando-se no currículo oficial de biologia. Desta vez eles desenvolveram seus próprios testes para indicar o conhecimento adquirido. Sem a ajuda de outros adultos ou crianças mais competentes, as crianças que trabalham em grupos, sem supervisão, no computador do projeto conseguiram aumentar seus escores nos resultados dos testes realizados de 7% para cerca de 30% em 75 dias, equiparando-se a outra escola de uma região rural próxima. Com mais 75 dias, com uma mediadora leiga, conseguiram aumentar sua pontuação para cerca de 51%, equiparando-se aos resultados encontrados em uma escola particular privilegiada em Nova Deli. A mediadora não tinha conhecimentos de biologia sobre o que estava sendo ensinado, e, por

isso, não conseguiria exercer efetivamente a função de ensinar os conteúdos. A sua função era incentivar as crianças, como fazem os avós, mencionando frases como “eu gostaria de poder fazer isso!”; “como diabos você descobriu isso?”; “eu nunca poderia ter entendido isso” ou “por favor, explique isso de forma simples para mim, eu tenho muito medo da ciência”.

3.1.5 Aprender com computadores

Com a popularização dos computadores pessoais, a partir dos anos 80 do século passado, novas possibilidades se abriram para o ensino. Se no início era uma questão de escolha, agora é fato que os computadores são uma realidade que a escola e os professores não podem ignorar. Turkle (1997) refere que a partir dos anos de 1970, começamos a ensinar nas escolas e universidades sobre computadores. Na concepção da época, ensinar sobre computadores seria ensinar a programar para que se pudesse dominar a máquina. Nesses tempos o computador era comparado a uma calculadora e o ensino dos algoritmos seria a chave da explicação do seu funcionamento, consequentemente a maneira de se aprender sobre computadores. A partir dos anos de 1980, com a proliferação das interfaces gráficas e das metáforas de ícones que representam documentos e programas, inicia-se uma concepção de simulação. Nesta nova fase não é preciso entender de algoritmos e números binários para se utilizar um editor de texto, planilha ou manipular documentos de forma geral. Sherry Turkle define, assim, duas visões da nossa sociedade sobre a alfabetização informática ou a aprendizagem com computadores.

Um grupo do Massachusetts Institute of Technology, de Papert a Resnick, passando por Alan Kay, estaria posicionado no grupo a que Turkle chama de geração da calculadora ou “Cultura do Cálculo”. Para este grupo é importante saber o que se passa em níveis mais profundos, abaixo da interface com o usuário, conhecendo os algoritmos e as estruturas da máquina. A programação de computadores aparece como foco central desse processo. Nesta corrente de pensamento não se deve simplesmente usar os aplicativos, mas ser capaz de alterá-los de forma a resolver problemas, chegando perto da epistemologia da ciência, como defende o construcionismo.

A outra corrente de pensamento é a que Turkle denomina de “Cultura da Simulação”. Para esta corrente, pouco importa como o computador resolve os problemas do usuário, o importante é que resolva. Na cultura da simulação, o foco da aprendizagem está nos aplicativos. Para esta corrente, não nos afastamos da realidade com a simulação, muito pelo contrário,

podemos fazer com que o usuário, ou aluno, experimente outras realidades, até mesmo às que dificilmente ele teria acesso. A meu ver, David Williamson Shaffer e James Paul Gee (Shaffer & Gee, 2012), ao proporem a utilização de jogos eletrônicos na educação, estariam nesta segunda corrente. É certo que, para experimentar os efeitos da gravidade e o movimento parabólico da física newtoniana, não se precisa de um jogo, mas jogar o Angry Birds engaja o aluno ao experienciar a física no jogo. O jogo, como uma forma de simulação, nos permite experimentar o que acontece quando, por exemplo, a velocidade se aproxima da velocidade da luz, o que seria impossível ou impraticável nas aulas tradicionais do ensino básico (Turkle, 1997).

Turkle também chama a atenção para o fato de a simulação ser aceita sem questionamento das regras escolhidas pelos projetistas dos aplicativos, levando a aceitação de um modelo de mundo único, induzido pelo *software* amigável. Turkle apresenta o caso de uma menina do décimo ano, que provavelmente aprendeu mais sobre urbanismo em duas horas, jogando SimCity, do que aprenderia lendo um livro. No entanto, comentando sobre o jogo ela é taxativa: "aumentar os impostos sempre leva a tumultos". Não passa pela cabeça da menina a possibilidade de que aumentar os impostos poderia melhorar a produtividade e o bem estar social, pois ela não questiona as regras e, principalmente, não se vê como uma pessoa capaz de modificá-las. De acordo com Sherry Turkle este efeito seria igual ao de uma pessoa que lê um livro mas não sabe o que significa.

Alan Kay observa que a educação está muito voltada para o mercado de trabalho, esquecendo a parte mais importante, que é formar cidadãos com experiência para debater ideias e pontos de vista, de modo a participar plenamente na democracia. Em um mundo sujeito a tantas mudanças, o conhecimento da ciência básica se faz necessário, segundo Kay, para que se possa ter mais discernimento e decidir mais sabiamente. Noções de ciência e de como ela funciona se tornaram necessárias para o nosso tempo, por motivos profissionais, por razões de cidadania, por razões de saúde e de nutrição, por razões artísticas e para nos tornarmos pessoas totalmente investidas no século XXI. Faz-se necessário saber como este conhecimento científico é adquirido, criticado e reformulado (Kay, 2003). Kay aponta a dicotomia de duas culturas criadas no século XX, uma voltada para a literatura e outra para as ciências e a matemática. Critica a formação na área de ciências e matemática como estando aquém das necessidades da sociedade atual. Menciona que é mais fácil achar alguém de ciências que tenha uma base literária do que um literato que tenha uma base de ciências (Kay, 2003).

3.2 Como se aprende Matemática?

É recorrente na literatura que matemática aprende-se resolvendo problemas e com uma reflexão a respeito dos problemas (Smole & Diniz, 2001; Zulatto, 2007).

> No âmbito da aprendizagem matemática, a interação também é condição necessária no seu processo. Trocar ideias, compartilhar soluções de problemas, expor o raciocínio é parte do 'fazer' matemática. Essa possibilidade tem sido pouco utilizada. Embora a troca interativa promova a aprendizagem significativa, fruto da participação ativa do aluno na produção do conhecimento, o que se percebe é que diálogo, comunicação em grupo e interação são valorizados por correntes educativas contemporâneas, mas pouco incorporados aos processos educacionais. (Zulatto, 2007, p. 52)

Rúbia Zulatto afirma que a aprendizagem da matemática se dá com a interação e o diálogo, produzindo novos significados em um processo colaborativo (Zulatto, 2007). A interação e o diálogo estimulam no aluno o raciocínio, a argumentação (discussão) e a visualização.

Segundo o filósofo e lógico americano Charles Sanders Peirce, existem três tipos de raciocínio: indutivo, dedutivo e abdutivo. O processo dedutivo se faz do geral para o particular, o processo indutivo das particularidades para a generalização e o abdutivo é uma combinação dos dois processos anteriores, acrescido da manipulação de conceitos de modo a testar conjecturas matematicamente (Zulatto, 2007). Sobre abdução, Zulatto escreveu:

> Entendo que indução, dedução e abdução podem atuar de forma articulada e condicionam a produção de conhecimento matemático, em um processo de levantar hipóteses, procurar testá-las e justificá-las para verificar sua veracidade. (Zulatto, 2007, p. 72)

Estimular a discussão e a argumentação é uma maneira de aprender matemática. A prova e a demonstração são temas recorrentes na matemática, porém, para a educação matemática o mais importante é a argumentação (Zulatto, 2007). Para conseguir a discussão produtiva devemos negociar normas que Boavida chama de sociais e sociomatemáticas, exercitando o respeito pela fala do outro e a audição ativa exercida pelo aluno como parte do seu processo ativo de aprendizagem. Para que a argumentação ocorra é preciso dar espaço ao aluno para expor seu ponto de vista. Deve-se ter em conta o outro, seja argumentando ou ouvindo. A argumentação e a troca de ideias é o que fica em evidência nas discussões.

> Um outro aspecto que se destaca como particularmente relevante para a argumentação matemática é a negociação de *normas sociais* e *normas sociomatemáticas* que colocam a ênfase na expressão audível, na escuta atenta, na partilha de ideias, na manifestação pública de desacordos e na explicação e justificação de contribuições. Atributos do processo de negociação cuja conjunção parece ser significativa para ajudar os alunos a

> apropriarem-se destas normas, são a *importância da sistematicidade e persistência*; a *pertinência de uma negociação contextualizada*; e a *essencialidade da coerência*. No seu conjunto, estes atributos remetem para a necessidade de no processo de negociação existir uma forte e sistemática consistência entre o que explicitamente se diz e as mensagens que implicitamente se veiculam através do modo como se age. (Boavida, 2005, p. 6)

Segundo Ana Boavida, a argumentação matemática é o processo de argumentação desenvolvido na aula de matemática, assumindo sua forma ao explicitar raciocínios e justificativas com o propósito de diminuir o risco e a incerteza de se enveredar por um caminho difícil ou errado de raciocínio. A argumentação serve para convencer uma audiência a aceitar ou rejeitar ideias, enunciados ou posições em função da razão. Embora a argumentação se sirva da linguagem natural entre o argumentador e o interlocutor, ela não se limita ao discurso, podendo incluir figuras, dados numéricos ou algébricos (Boavida, 2005). As teorias de Vygotsky e Bruner enfatizam a importância da linguagem e do convívio social na aprendizagem, restando aos educadores matemáticos discutir qual a melhor forma de se estabelecer esta comunicação para se comunicar expressando ideias matemáticas. Segundo Boavida (2005), a comunicação tem tido o consenso da academia, colocando a argumentação no centro dos processos discursivos e quebrando assim a ideia de que a ciência é a expressão da verdade. De fato, ela é construída pelo processo de argumentação e convencimento dos pares, tanto quanto o uso do método científico.

A prova utilizada pela matemática aparece como caso particular da argumentação. A prova tem suas restrições, necessitando da explicitação do raciocínio no produto final, enquanto a argumentação pode fazer uso de analogias, metáforas e outros processos típicos da abdução e do fazer matemática. Sendo assim, a argumentação pode ser um facilitador para se ensinar a prova matemática e a educação para a argumentação é mais que um educar para a matemática. É um educar para a democracia, contribuindo pelo lado social e ético, indo além da educação intelectual (Boavida, 2005).

A importância da argumentação é apresentada por Ana Boavida em uma lista de motivos que se complementam e interagem:

> (a) a valorização do raciocínio matemático nas suas múltiplas vertentes numa perspectiva que não põe a ênfase no rigor e formalismo entendidos como um fim em si mesmo,
> (b) a recomendação de que os alunos aprendam matemática com compreensão,
> (c) o valor atribuído às linguagens naturais e à interação social para a aprendizagem,

(d) a aproximação da comunicação na aula de matemática da existente na comunidade dos matemáticos,
(e) dificuldades encontradas na aprendizagem da prova e a procura de caminhos que facilitem esta aprendizagem e
(f) a relevância de a escola proporcionar a todos os alunos condições necessárias para desenvolverem certas competências transversais, entre as quais está a competência argumentativa, fundamentais ao exercício pleno de uma cidadania responsável numa sociedade democrática. (Boavida, 2005, p. 7)

A visualização não se limita ao sentido físico da visão, mas ao intelecto, à compreensão e à intuição matemática. Os alunos devem ser capazes de entender e representar ideias de forma a visualizar e fazer o outro visualizar os conceitos.

3.2.1 Práticas criticadas e desafios para a mudança

O Movimento da Matemática Moderna se deu em todo o mundo na década de 1960. Ele tinha a intenção de modernizar o ensino de matemática da época frente às tecnologias emergentes. Conteúdos, como estruturas algébricas, teoria dos conjuntos, topologia, transformações geométricas foram incluídas nos currículos.

A implantação da matemática moderna foi realizada em todo o Brasil em épocas, níveis e escolas diferentes (Wielewski, 2008). Embora tivessem as melhores intenções, os pesquisadores pós anos 80 apontam algumas práticas danosas que se estabeleceram, apesar, ou à revelia, dos esperançosos pioneiros. Estas reações guiaram às atuais tendências do ensino de matemática (Hoff, 1996; Zorzan, 2012).

A concepção da matemática que se estabeleceu foi a de um conhecimento pronto e formalizado, de verdades definitivas, infalíveis e imutáveis. Desde sempre a matemática foi mostrada como um conhecimento neutro, era um produto acabado, sem uma raiz cultural, onde o erro é ignorado. Uma mesma matemática é ensinada em diferentes classes, de uma mesma série, em diferentes escolas ao redor do mundo (Hoff, 1996). Em termos metodológicos, o professor era o detentor do conhecimento desta matemática, pronta, final e acabada e deveria passar estes conhecimentos abstratos e formais aos alunos, por um processo de transferência de informação. Para isso, eram explicados os conceitos, resolvidos exercícios padrão e fixados, através da repetição e reprodução da técnica, em exercícios. A matemática pronta é o ponto de partida da elaboração didática e o ponto de chegada do aluno.

As aulas são estruturadas logicamente a partir dos conteúdos matemáticos já formalizados e a estrutura da própria disciplina, sem levar em conta o processo de elaboração

do conhecimento do aluno. Ao aluno cabe memorizar, assistindo passivamente as demonstrações e aplicar e reproduzir as soluções entregando as respostas rapidamente e sem erros. O pensar criticamente e autonomamente não é estimulado. Os conteúdos matemáticos são dados de forma fragmentada e isolados de outros conteúdos matemáticos. Aparentemente, como destaca Hoff (1996), não se relacionam com outras disciplinas, não é utilizada a matemática para buscar soluções e resolver problemas reais. Investir tempo para que o aluno recrie relações já estabelecidas por outros é visto como perda de tempo, uma vez que a matemática já está posta, sendo o ensino tanto melhor quanto mais rápido for o processo de transmissão direta dos conteúdos abstratos.

Segundo diversos autores, este modelo de ensino desrespeita as etapas necessárias à estruturação do conhecimento por parte do aluno. Sob esta concepção, o aluno é visto como um ser desprovido de ideias e de conceitos, cabendo ao professor inseri-los de fora para dentro. Nesse modelo não se leva em conta a matemática usada e desenvolvida pelos alunos em outros contextos que não sejam o escolar. Os indicadores de baixo rendimento em matemática eram explicados pelo mau desempenho dos alunos, isentando os professores e o sistema do fracasso da escola. Também cabia à escola a divulgação da crença de que a matemática era para poucos, para *dotados* de inteligência. Os cursos de formação de professores reafirmavam este modelo de uma matemática acabada e a metodologia da transmissão e memorização, realimentando este ciclo vicioso. A crença de que o aluno é uma tábua rasa, vinda do empirismo, e a posição apriorística (determinista) de que inteligência e aprendizagem são dons de alguns privilegiados geneticamente, também dificultam as mudanças. Dentro dessa lógica, alguns chegam a "saber" matemática, enquanto a grande maioria deve-se contentar com o "saber fazer" do automatismo. (Hoff, 1996).

Do ponto de vista político, esta metodologia favorece o autoritarismo e a submissão passiva, com ausência da crítica. Esta matemática tem formado pessoas desqualificadas e não questionadoras, estando, assim, despreparadas pra serem agentes transformadores sociais. Quanto à formação de valores e atitudes ensinadas por esta política pedagógica, "é como se o professor estivesse dizendo ao aluno (...) na matemática, como em tudo na vida, as regras já estão estabelecidas para sempre e só nos resta aceitar (...) sem nunca perguntar por que elas são assim" (Hoff, 1996, p. 77).

Miriam Hoff mostra como o modelo desenvolvimentista brasileiro pós-1964 se servia dessas concepções, ao defender uma ideologia tecnicista que privilegiava o ensino

profissionalizante, formando especialistas eficientes em funções determinadas, mas sem que estes especialistas questionassem o processo como um todo ou mesmo o sistema político-social mais amplo, ao qual ele estava servindo (Hoff, 1996).

3.2.2 Tendências em Educação Matemática

Em reação ao quadro diagnosticado por pesquisadores em educação matemática, várias iniciativas de oposição ao *status quo* foram desenvolvidas para o ensino de matemática, conhecidas como Tendências em Educação Matemática. Estas reações contra-hegemônicas podem ser classificadas sob quatro aspectos: mudança epistemológica; considerações quanto aos processos cognitivos do aluno; considerações quanto a matemática e a cultura; considerações quanto a renovação do ensino de matemática (Hoff, 1996).

Do ponto de vista epistemológico, as posições empiristas e apriorísticas são deixadas a favor de uma visão interacionista, com forte influência teórica de Jean Piaget e Emilia Ferrero e de autores marxistas como Lev Vygotsky e Paulo Freire. Assim, acredita-se que o conhecimento é construído pelo indivíduo em interação com o seu meio à medida que vai construindo internamente sua própria estrutura cognitiva (Hoff, 1996).

Do ponto de vista dos processos cognitivos, os professores mudam seus métodos em função desse novo referencial teórico. Eles devem considerar os conhecimentos prévios do aluno, sem querer colocar um mecanismo de relações preconcebidas ao aluno, mas sim conduzi-lo a refazer as suas relações internas, integrando os novos conhecimentos, em um processo que é mental e interno a cada indivíduo. Segundo Hoff (1996), a aula é estruturada para facilitar estes processos internos, numa lógica cognitiva, e não é estritamente baseada na lógica interna da disciplina. Os erros passam a ser vistos como parte do processo cognitivo. Do ponto de vista do relacionamento entre matemática e cultura, considera-se a relação dos alunos com a matemática fora da escola, devido às suas vivências e necessidades. As considerações com relação à renovação do ensino de matemática refletem-se na postura do professor em três aspectos: o viés político, o enfoque dinâmico do conhecimento e a postura de professor-pesquisador.

Quanto ao viés político, o professor deixa de se dizer apolítico, acreditando que mesmo esta postura se coaduna com uma postura política vigente. Acredita-se que, em sala de aula, não se pode estar restrito a ensinar e aprender os conteúdos de matemática, mas este processo faz parte de uma educação mais ampla que se reflete nos valores e atitudes sobre a realidade

social. O professor passa a identificar a educação como um ato político reconhecido e assumido e acredita que ao ensinar e aprender matemática é necessário contribuir para a educação crítica e não alienante. O enfoque dinâmico do conhecimento parte do pressuposto de que a matemática não é um produto acabado e imutável, mas sim o produto de um processo histórico. Acreditando que o aluno é um sujeito ativo, que constrói seu conhecimento, o professor passa a ser um facilitador da aprendizagem. O professor usa seus conhecimentos para problematizar situações que contribuam para elaboração do conhecimento por parte do aluno.

Neste contexto emerge o professor-pesquisador. O professor, partindo de uma visão dinâmica do conhecimento, deve tomar sua própria prática como ato contínuo de aprendizagem, gerando novos conhecimentos. O professor, por meio da pesquisa, deve submeter sua prática a um acompanhamento crítico refletindo e realimentando o fazer docente.

Entre as diversas tendências em educação matemática, que emergiram desses novos paradigmas, gostaria de mencionar quatro delas que me parecem bem conhecidas dos educadores matemáticos e dialogam com minha investigação. São elas: Resolução de Problemas, Etnomatemática, Modelagem e Tecnologias.

3.2.2.1 Perspectiva da Resolução de Problemas

A tendência Resolução de Problemas na educação matemática pretende incentivar a compreensão e a iniciativa da busca pela solução de uma situação-problema. Já em 1896 e 1904, Dewey enfatizava a resolução de problemas e a prática de projetos como orientação pedagógica (Zorzan, 2012). Porém, foi em 1944, com a publicação do livro *A arte de Resolver Problemas,* de George Pólya, que a resolução de problemas foi usada como uma forma de ensinar matemática (Onuchic & Allevato, 2011). Em 1980, o National Council of Teachers of Mathematics (NCTM) publicou o documento *An Agenda for Action: Recommendations for School Mathematics in the 1980's*, indicando que a "resolução de problemas deve ser o foco da matemática escolar". As ideias eram fundamentadas pelo construtivismo e tinha a teoria sócio-cultural de Vygotsky como teoria principal. A partir desse documento, três abordagens na utilização da resolução de problemas se apresentaram segundo a concepção de diferentes grupos. As abordagens são (1) ensinar sobre resolução de problemas; (2) ensinar matemática para resolver problemas; e (3) ensinar matemática através da resolução de problemas.

Os seguidores de Pólya entendiam que se devia teorizar a resolução de problemas ensinando o método e as estratégias e defendiam o "ensinar sobre resolução de problemas".

Outros defendiam que se deveria ensinar a matemática formal e depois colocar um problema para ser resolvido, sendo assim, devia-se "ensinar matemática para resolver problemas" (Onuchic & Allevato, 2011). A perspectiva de "ensinar matemática através da resolução de problemas" argumenta que, didaticamente, o ensino-aprendizagem ocorre a partir de uma situação-problema passando pelos processos de problematização para o estudo abstrato, representando e operacionalizando os problemas de forma simbólica (Zorzan, 2012).

Muitas vezes se confunde o problema com exercícios de aplicação utilizados para fixar a teoria apresentada. Neste caso, o exercício está dentro do paradigma de transmissão e memorização. O aluno, ao fazer este exercício, pratica apenas a imitação, a repetição e a memorização. O problema utilizado na tendência de resolução de problemas está em uma situação-problema na qual o aluno não sabe a priori o resultado nem o processo da resolução. Para esclarecer, vou apresentar o exemplo da Tabela 4, que consta do curso "Programa Gestão da Aprendizagem Escolar" do Ministério da Educação do Brasil (Dias, 2008).

Tabela 4 - Exercício x Situação-Problema (Dias, 2008)

Exercício	Exercício O pai do Marco tem um terreno retangular de 6m por 9m e resolveu cercá-lo com uma rede de arame, cujo metro custa R$37,00. Para fixá-la, vai precisar de 10 estacas a R$10,00 cada uma. Quer também colocar um portão de 1m de comprimento, cujo preço é R$570,00. Qual é o preço total desses materiais? Características: • fornece todas as informações necessárias; • não dá informações supérfluas; • o aluno deve usar conceitos matemáticos; • o aluno deve combinar os dados do problema por meio de operações conhecidas; • a resposta ao problema é um único número

Situação-problema	Situação-problema O pai do Marco tem um terreno retangular de 6m por 9m e resolveu cercá-lo com uma rede de arame, que necessita de estacas para fixação. Ele quer também colocar um portão. Procurando materiais, ele encontrou: • rede de arame a R$37,00 o metro, que necessita de postes para fixação, colocados de 1,5m em 1,5m. • rede de arame a R$51,00 o metro, que necessita de postes para fixação, colocados de 3m em 3m. • portão com 1m de comprimento, por R$570,00. • portão com 1,5m de comprimento, por R$750,00. • estacas, a R$10,00 cada uma. • arame para amarrar a rede a cada estaca, a R$2,00 o metro. É necessário 1m de arame para amarrar as redes a cada estaca. • tinta para pintura da casa, a R$40,00 o galão. O pai do Marco quer saber o que deve comprar para fazer a cerca com o portão e gastar o mínimo possível. Características: • requer o estudo de várias hipóteses: a) cerca mais barata com portão mais barato (porém mais estacas e mais arame); b) cerca mais barata com portão mais caro (porém mais estacas e mais arame); c) cerca mais cara e portão mais barato (porém menos estacas e menos arame); d) cerca mais cara com portão mais caro (porém menos estacas e menos arame); • requer estudo da colocação das estacas; • requer decisões – o número de estacas numa lateral nem sempre dá exato. O que fazer: colocar uma estaca a mais ou uma a menos? • pode haver respostas distintas, dependendo da opção proposta no item anterior; • o problema tem dado supérfluo – o preço da tinta.

3.2.2.2 Perspectiva da Etnomatemática

A tendência etnomatemática teve seu início na década de 1970 e enfatiza a matemática informal e, segundo Ubiratan D'Ambrósio, que cunhou o termo etnomatemática, é mais do que uma matemática de etnias. Mas este termo deve ser compreendido a partir das raízes *etno*, *matema* e *tica*, significando que há várias maneiras, técnicas, habilidades (*ticas*) de explicar, de entender, de lidar e de conviver (*matema*) com distintos contextos naturais e socioeconômicos da realidade (*etnos*), como refere (D'Ambrosio, 1996). Esta tendência abre a possibilidade da descoberta de várias matemáticas, tão variadas quanto os grupos étnicos. No Brasil, sociedades tão diversas quanto os povos indígenas, as comunidades quilombolas, os ribeirinhos da Amazônia, os sertanejos no nordeste, os gaúchos nas fronteiras do sul do Brasil, os pomeranos (que têm uma cultura viva aqui no Brasil e em extinção na Europa, de onde são originários), são apenas algumas das possibilidades de se descobrir maneiras inusitadas de se fazer, pensar e

viver matemática. A etnomatemática não quer ser um substituto da matemática acadêmica, mas pretende valorizar o saber matemático multicultural, tornando a matemática uma ciência do povo e valorizando o indivíduo como sujeito da história. A etnomatemática surge a partir da realidade e do contexto social local e, metodologicamente, é focada na interdisciplinaridade e na transdisciplinaridade, assim como da passagem do concreto para o abstrato (Zorzan, 2012).

3.2.2.3 Perspectiva da Modelagem Matemática

A tendência da modelagem matemática também valoriza a cultura do mundo vivido pelo aluno e também teve seu início na década de 1970. Adriana Zorzan aproveitou trabalhos de Maria Salett Biembengut e Nelson Hein, a partir dos quais propõe cinco passos para realizar um projeto de modelagem com os alunos:

> 1. Diagnóstico: da realidade, dos interesses dos alunos e do grau de conhecimento dos mesmos.
> 2. Escolha do tema ou modelo matemático: para desenvolver o conteúdo programático que estará inserido numa situação problemática.
> 3. Desenvolvimento do conteúdo programático: ocorre o reconhecimento da situação-problema, formulação e resolução do problema e interpretação e validação a partir do conteúdo.
> 4. Orientação de modelagem: requer que o sujeito seja capaz de fazer modelos matemáticos. O aluno é incentivado à pesquisa, a desenvolver a criatividade e a habilidade de formular e resolver problemas e a aplicar o conteúdo matemático. Nesse processo, o aluno é conduzido à formulação de hipóteses, à constituição de alternativas para solucionar as situações-problema.
> 5. Avaliação do processo: avaliam-se a produção e o conhecimento matemático, a produção do trabalho de modelagem em grupo e a extensão e aplicação do conhecimento para, assim, redirecionar o trabalho. (Zorzan, 2012, p. 82)

A modelagem coloca no centro da questão o conteúdo eleito pelos alunos que deve ser mediado, explorado, discutido e analisado com e através da matemática. O professor deve utilizar as suas competências técnicas e políticas para nortear o tema e os conteúdos abordados. A matemática, nesta tendência, é uma linguagem para solução de problemas reais estimulando a criatividade, lendo e interpretando a realidade (Zorzan, 2012). Esta tendência está baseada na interdisciplinaridade, possibilitando o aprofundamento em vários saberes, de uma forma descentralizada, exploratória e participativa. De forma crítica, a modelagem propõe transformações no contexto no qual vive o aluno (Zorzan, 2012).

3.2.2.4 Perspectiva da Tecnologia

A tendência tecnológica do ensino de matemática é uma preocupação da comunidade acadêmica desde os anos de 1980, acompanhando o crescente avanço tecnológico e o maior acesso a computadores e a informática de forma geral. Ao princípio, os professores se mostravam reticentes com relação ao uso das tecnologias temendo serem substituídos por ela. Mas, no século XXI, a tecnologia faz parte da vida e da cultura atual e a falta da perspectiva tecnológica pode resultar na exclusão social dos indivíduos, a exclusão digital para a qual nos alerta Zorzan (2012).

A adoção da tecnologia não é apenas uma adição de recursos, mas interfere nos processos pedagógicos. A tecnologia deixa obsoleto o método de transmissão e memorização, demandando uma outra postura por parte dos professores que insistem neste paradigma. Demanda o desenvolvimento do pensamento sistêmico, do conhecimento e da produção cultural. Como salienta Zorzan (2012), a produção matemática também deve acontecer a partir dos mais variados recursos tecnológicos, incluindo o computador, a Internet, a calculadora, o celular etc.).

Em termos metodológicos, a tendência tecnológica deve acontecer de forma interdisciplinar, sob o enfoque da modelagem e através de trabalhos cooperativos e colaborativos. O aluno não deve ter como foco memorizar o que se encontra em grande quantidade na Internet nem outras formas artificiais de expansão da memória, mas deve saber como lidar com tamanha quantidade de informação. O aluno deve saber como procurar as informações com razoável grau de confiabilidade, deve saber interpretá-las, analisá-las e reconstruí-las. A tendência do ensino de matemática na perspectiva tecnológica tem como objetivo estimular a curiosidade, a imaginação, a comunicação em diferentes mídias, a criação de soluções para os problemas e o desenvolvimento das capacidades cognitivas, afetivas e sociais (Zorzan, 2012).

3.3 Como se aprende programação de computadores?

Roy Pea e Midian Kurland (1984) listam sete argumentos dos mais utilizados para se justificar o ensino de programação de computadores:

(1) o pensamento rigoroso, expressão precisa, necessidade de fazer suposições explícitas para que o computador execute o algoritmo desejado;

(2) a compreensão de conceitos gerais, como procedimento formal, variável, função e transformação;

(3) o desenvolvimento de habilidades “heurísticas”, utilizadas na resolução de problemas em qualquer domínio, tais como planejamento, encontrar um problema relacionado, resolvendo o problema, decompor um problema em partes etc.;

(4) a ideia geral de que a depuração de erros é passível de planejamento e aplicável a qualquer tipo de resolução de problemas;

(5) a ideia geral de que se pode inventar pequenos procedimentos, como blocos de construção, para se ir construindo gradualmente soluções para grandes problemas;

(6) o reforço da conscientização da alfabetização sobre o processo de resolução de problemas devido à prática de discutir o processo de resolução de problemas em programação por meio da linguagem de conceitos de programação;

(7) um maior reconhecimento de domínios para além da programação em que raramente há um único caminho para fazer algo. Diferentes formas têm custos e benefícios diferentes.

Pea e Kurland também criticam dois mitos sobre o ensino de computação. O primeiro é o de que aprender os comandos e a sintaxe da linguagem de programação é suficiente para aprender a programar. O segundo mito é o de que as habilidades de resolução de problemas em programação são espontaneamente repassadas a outras áreas do conhecimento (Pea & Kurland, 1984). Porém, Akcaoğlu afirma que se pode ensinar a resolução de problemas usando projetos de software, desde que se realize um planejamento curricular com este fim (Akcaoğlu, 2013). Pea e Kurland anotam também as principais restrições para o aprendizado de programação, incluindo a habilidade matemática, a capacidade de memória, a capacidade de raciocínio analógico, as habilidades de raciocínio condicional, as habilidades de pensamento procedural e as habilidades de raciocínio temporal (1984). Vê-se, por esta lista, mais uma vez, como as competências matemáticas e do pensamento computacional são recorrentes na programação de comutadores.

3.3.1 A importância do algoritmo

Marcus Maltempi e José Armando Valente definem a programação de computadores como uma atividade de resolução de problemas que exige criatividade, uma vez que é necessário conhecer o problema e criar um algoritmo que leve à resolução (Maltempi & Valente,

2000). Acrescento ainda que o programador deve tomar decisões para escolher uma sequência de comandos, já que existe mais de uma sequência que pode levar ao mesmo resultado computacional. Além de decisões sobre a forma de apresentação ao usuário e da abordagem do problema no nível conceitual, Maltempi e Valente acrescentam que, diferentemente da maioria das resoluções de problemas propostos na escola, a programação exige a explicitação da solução em uma linguagem formal de programação. Esta necessidade de formalismo pode-se observar em outras áreas, como a matemática e as ciências em geral. Por exemplo, o formalismo matemático encontra-se na legislação. O que é o nosso conjunto de leis senão uma máquina de inferências para definir as regras de convivência?

Programar segue um círculo de atividades descrição-execução-reflexão-depuração (Maltempi & Valente, 2000). Ao definir o conjunto de comandos e variáveis que vão compor o programa, o programador está descrevendo a solução. Depois disso, executa o programa e observa os resultados. Após uma reflexão a respeito do resultado, o programador pode se sentir satisfeito com o resultado ou, na maioria das vezes, realiza a atividade de depuração para corrigir os erros ou melhorar o resultado obtido. Sendo assim, ao programar, o aluno cria um modelo mental e formaliza este modelo na linguagem de programação. A partir daí ele pode fazer experiências com o formalismo e aprender com seus erros, resolvendo problemas, concebendo soluções em etapas e tomando decisões. Estas competências, exercitadas na atividade de programação, caracterizam o pensamento computacional.

Por princípio, qualquer linguagem de programação é boa para descrever o problema de maneira formal, mas as linguagens que facilitam esta interface humano-computador são mais interessantes, porque o programador pode se dedicar à resolução do problema, mais do que aos detalhes da linguagem computacional, como sugere José Armando Valente.

> As linguagens para representação da solução do problema podem, em princípio, ser qualquer linguagem de computação, como o BASIC, o Pascal, ou o Logo. No entanto, deve ser notado que o objetivo não é ensinar programação de computadores e sim como representar a solução de um problema segundo uma linguagem computacional. O produto final pode ser o mesmo — obtenção de um programa de computador — os meios são diferentes. Assim, como meio de representação, o processo de aquisição da linguagem de computação deve ser a mais transparente e a menos problemática possível. Ela é um veículo para expressão de uma ideia e não o objeto de estudo. (J. A. Valente, 1993, p. 14)

Neste aspecto, o Scratch aparece como uma boa escolha, por facilitar a programação com comandos e estruturas as quais se encaixam como um quebra-cabeças, evitando os erros

de digitação e sintaxe que ocorrem frequentemente, quando se utilizam os compiladores tradicionais. O programa deve ser a descrição formalizada de uma ideia e um meio de se expressar.

3.3.2 Abordagem construtivista da aprendizagem

Papert se baseou no construtivismo de Jean Piaget, acreditando que os aprendizes arquitetam suas próprias estruturas intelectuais construindo seu próprio conhecimento. Papert observou que esta construção do conhecimento acontece naturalmente no contexto da construção de algo externo, que se possa mostrar e compartilhar com outros (Lima, 2009). A esta espécie de construtivismo, na qual se constrói um objeto externo e compartilhável, Papert chamou construcionismo. As linguagens de programação como o Logo e o Scratch dão a possibilidade de construção de objetos computacionais abstratos, mas igualmente externos ao autor e compartilháveis.

O construcionismo propõe a construção de um artefato, isto é, a construção de entidade pública. O conhecimento acontece no processo de construção, aprendendo enquanto faz. Segundo Papert, este objeto pode ser tão diverso como um castelo de areia ou uma teoria sobre o universo. Durante a construção, os conceitos necessários são formulados. Das diversas formas possíveis de se fazer emergem as diversas formas de entender o conceito. O construcionismo privilegia os caminhos individuais ao invés do melhor caminho. Neste aspecto, é melhor do que o instrucionismo que mostra o melhor caminho, ou o caminho adotado pela ciência. Na fase de formação de conceito é melhor que se mantenha a mente aberta até que o aluno formule e reformule o seu próprio conceito, assim as diferenças são respeitadas (Papert, 1991).

Papert lembra as aulas de artes que ele via na escola, na qual os alunos faziam escultura em sabão. O projeto das esculturas não se resumia a uma aula, mas se estendia por semanas. Neste processo tem-se tempo para pensar, para sonhar, para contemplar, para obter uma nova ideia e experimentar, desistir, persistir; tempo para conversar, para ver o trabalho do outro e a reação do outro ao seu trabalho. Antes de formular o conceito de construcionismo ele sabia que queria que as aulas de matemática fossem assim, como as aulas de artes. Depois, conta que isso foi possível com o Logo, em diversos exemplos que destaca. Por exemplo, o caso do aluno da 5.ª série que fez uma espetacular tela gráfica através de programação. Segundo o aluno, “eu tive que imaginar ângulos e curvas para obter a melhor beleza na figura”. O aluno se apropriou

dos conceitos matemáticos de forma bem pessoal, sem separar matemática de imaginação (Papert & Harel, 1991).

Papert fala de dois estilos de fazer as coisas. Um é o clássico da programação estruturada, que vai do geral para o particular. Partindo de um plano abstrato, fazendo o plano tomar forma no mundo real. Outro representa aqueles que sempre querem um objeto palpável, no qual modificam até chegar ao desejado, usando o objeto como apoio ao modo de pensar. Segundo Papert, esta última forma é predominantemente preferida pelas meninas. Papert observa a impossibilidade de entender por completo por que este processo construtivista funciona. Por outras palavras, como acontece o aprendizado, mas independentemente do porquê do seu funcionamento, podemos lançar mão desse fato para o ensino, agregando as novas tecnologias.

> Construir e brincar com castelos de areia, família de bonecas, casas de Lego e coleções de figurinhas proporcionam imagens de atividades bem enraizadas na cultura contemporânea e é plausível que no processo de aprendizagem se vá para além dessas limitadas habilidades. Eu não acredito que alguém entenda completamente o que dá a estas atividades a qualidade de um 'rico aprendizado'. Mas isto não impede que se tome como modelo na presença de novas tecnologias para expandir o escopo dessas atividades com esta qualidade. (Papert & Harel, 1991, p. 6)

Tomando a matemática como exemplo, Papert diz que, ao construir modelos os alunos ficarão interessados em aprender e aprenderão mesmo com pouca ou nenhuma orientação do professor. Isso levaria a uma mudança mais profunda no processo de aprendizagem do que mudar apenas as possibilidades de ensinar, propostas pelos programas instrucionistas que independem do professor. Papert diz que, com o construcionismo, se promove um ambiente de aprendizagem em micromundos nos quais as pessoas aprendem.

A discussão entre construcionismo e instrucionismo não é meramente uma disputa entre duas maneiras de adquirir conhecimento. No instrucionismo, o foco recai sobre os métodos de ensinar, enquanto no construcionismo trabalha-se com a natureza do conhecimento, levando o aluno a questionamentos relativos à filosofia e à epistemologia do conhecimento. Levando a conceber que o pensamento não é apenas livre nas ideias, mas também na forma de pensar, como acontece com movimentos sociais, como o feminismo, o africanismo. Eu acrescentaria as tendências da etnomatemática, as etnociências e os atuais movimentos de educação do campo, que reivindicam uma educação não urbanocêntrica, proposta pelos campesinos para campesinos, levando em consideração suas culturas locais. Segundo Papert, o instrucionismo e

o uso de programas baseados em Instruções Auxiliadas por Computador (CAI) perpetuam o autoritarismo e muitos hábitos criticados que se encontram nas escolas (Papert & Harel, 1991).

3.3.3 A importância dos projetos

O construcionismo de Papert propõe que o aluno vá formando seus próprios conceitos na medida em que constrói um artefato externo que se possa mostrar. Nesta prática, o aluno levará algumas aulas pensando e repensando seu projeto, discutindo com colegas, mostrando o que já fez e o que pretende fazer. A este propósito vale a pena observar o que nos diz Resnick (2007) a respeito dos projetos em seu artigo "Technologies for lifelong kindergarten". O investigador do MIT apresenta diversas razões a favor da adoção dos projetos como recurso didático:

- As atividades de projeto colocam os alunos como participantes ativos. Requer mais controle e envolvimento pessoal nos processos de aprendizagem, se comparadas com as atividades tradicionais de transmissão de conhecimento.
- As atividades de projeto são muitas vezes interdisciplinares, reunindo conceitos de artes, matemática e ciências.
- As atividades de projeto evitam o pensamento dicotômico (certo/errado) predominante nas atividades escolares, possibilitando várias estratégias e soluções.
- As atividades de projeto fornecem um contexto para a reflexão. Os artefatos construídos servem como sombras externas de modelos mentais internos dos alunos. Servem como referência para reflexão, revisão e ampliação dos modelos internos do mundo.
- As atividades de projeto incentivam as crianças a colocar-se na mente dos outros, já que precisam de pensar em como as outras pessoas vão entender e utilizar as suas construções.

Lygeia Ricciardi (Ricciardi, n.d.) conta de uma experiência de Papert passada no Centro de Jovens da cidade de South Portland, no estado do Maine nos Estados Unidos. Neste centro se encontravam encarcerados adolescentes de doze a vinte anos por crimes como roubo, vandalismo e até assassinatos. Neste ambiente Papert desenvolveu um experimento no qual os alunos desenvolviam projetos com tijolos de Lego, carros robóticos, boletins eletrônicos e videogames. O que Papert pretendia era que os adolescentes tivessem mais do que habilidades técnicas. Pretendia que eles tivessem uma visão positiva de si e do mundo. Eles deveriam se ver como contribuintes de um mundo que estamos criando e no qual viveremos. Para isso, os alunos devem aprender a se concentrar por períodos longos de tempo, a serem automotivados e

autoconfiantes. Papert combina materiais de alta tecnologia com materiais de baixa tecnologia como livros, blocos de construção, papel e canetas. Para Papert os computadores facilitam a aprendizagem, em parte porque eles se adaptam a vários estilos de aprendizagem. Os computadores possibilitam ainda a veiculação de conceitos poderosos na ciência e na matemática, como os relacionados com formas geométricas, variáveis, fórmulas, funções e gráficos. Programando computadores, fazendo simulações e construindo robôs, os jovens têm acesso a conceitos abstratos, como velocidade e torque, ao mesmo tempo em que desenvolvem suas habilidades de comunicação. Segundo Ricciardi, Papert gosta de citar a máxima de Albert Einstein: "o amor é um professor melhor do que o dever".

3.3.4 Promovendo o aprendizado de programação

Neste item foram agrupadas algumas recomendações encontradas na literatura, que podem ajudar o professor de maneira mais pragmática.

César Coll e colaboradores, no livro *O Construtivismo em Sala de Aula*, referem que o construtivismo pressupõe que, quando aprendemos, nos envolvamos globalmente na aprendizagem, incluindo, não apenas de conteúdos concretos, mas também de aspectos afetivos e relacionais. Por outro lado, mesmo que inconscientemente, ensinamos não apenas conteúdos, mas também atitudes, valores e normas (Coll et al., 2001). Os mesmos autores apresentam também algumas ideias de ordem prática relacionadas com as metodologias construtivistas em sala de aula, como segue:

- A exposição do conteúdo deve ser feita de maneira gradual sem grandes saltos ou rupturas com momentos de recapitulação, resumo e síntese.
- O professor deve fazer analogias, usando os conhecimentos prévios dos alunos.
- Deve-se mostrar de forma deliberada o que está sendo feito ou se pretende fazer com relação às atividades e ao que se quer ensinar.
- O aluno deve poder avaliar o seu próprio processo de aprendizagem.
- O aluno deve ter motivos relevantes para aprender os procedimentos e sentir-se satisfeito em aprendê-los.
- O aluno deve colaborar e contar com a colaboração de outros.
- O aluno deve tender a acreditar que a elaboração do próprio conhecimento depende de um esforço pessoal.

– Deve-se dar ao aluno a oportunidade de executar os procedimentos de forma voluntária, consciente e inovadora, bem como revisar a execução e realizar seus aperfeiçoamentos.

Estas ideias de ordem prática podem ser mais bem observadas nas palavras dos próprios autores:

> Os alunos que aprendem procedimentos (como a observação de plantas) precisam de conceitos (tipos e partes das plantas) e de desenvolver determinadas atitudes (desejo de saber, rigor, cumprimento das normas, entre outras). As atividades didáticas integram, geralmente, todos estes aspectos, embora haja um que é eleito como objetivo de aprendizagem em determinada situação concreta de ensino. Determinar o objetivo ajuda a orientar melhor a atividade do aluno e da aluna no processo de construção de conhecimentos, também permitindo que os professores decidam melhor o tipo e grau de ajuda a prestar. (Coll et al., 2001, p. 104)

Bruner considera fundamental a ação, a participação em atividades compartilhadas e a negociação de significados para um sistema cognitivo em construção (Correia, 2003).

3.3.5 Ambientes inovadores de aprendizagem

Luciano Meira e Marina Pinheiro (2012) notam que a escola mudou muito pouco nos dois séculos da sua existência. A tecnologia não é uma solução para mudar as relações estabelecidas na escola. Segundo os autores, a escola precisa se renovar e, para isso, propõem que se tomem como base 5 premissas para criar na escola cenários de aprendizagem. Estas 5 premissas formam o acrônimo D3NA. Usando o DNA biológico como metáfora, os investigadores afirmam que a escola não sofreu mutação nestes 2 séculos. O D3NA seria a mutação proposta para a inovação, definindo inovação como sendo uma novidade disseminada que provoca mudança de comportamento nas pessoas. D3NA se refere a “Diversão, Diálogo e Desafios bem balanceados, colocados numa estrutura Narrativa capaz de produzir uma Aventura educacional” (p. 3), melhor explicado na Tabela 5.

Tabela 5 – Explicação do Acrônimo D3NA

Premissa	Explicação
Diversão	Se refere à importância da ludicidade no processo.
Diálogo	Se refere às interações entre pessoas, de forma direta ou através de uma mídia como o livro ou uma rede social.
Desafio	Se refere à necessidade de um desafio adequado de modo a não causar frustração (desafio difícil demais) ou tédio (desafio fácil demais).
Narrativa	Se refere à capacidade de criar e entender narrativas como forma essencial para o processo e avaliação da aprendizagem. Meira comenta que até a aula de história tem se reduzido a uma lista de datas e fatos sem uma narrativa.
Aventura	Se refere À predisposição a aventura. Encarando a novidade e a incerteza de forma positiva.

Meira e Pinheiro relatam a experiência realizada no Estado de Pernambuco, Brasil, com as Olimpíadas de Jogos e Educação (OJE), uma plataforma web, para alunos e professores do 8.º ano do ensino fundamental até ao 3.º ano do ensino médio. Nesta plataforma alunos e professores colaboram em uma rede social *gamificada* para resolução de desafios. Os desafios são baseados nos conteúdos da matriz curricular do ensino médio. Jogos com mecânicas clássicas de tiros e naves espaciais, adaptados aos contextos educacionais: "por exemplo, torna-se desafio lúdico um jogo em que o usuário – representado por uma cápsula que viaja no sistema venoso – precisa distinguir as formas das estruturas maléficas à saúde de outros elementos do sangue, para atingir o sucesso de sua missão" (Meira & Pinheiro, 2012, p. 5). Estes Jogos são chamados de *mini games*, mas existem também os enigmas, que são questões baseados no Exame Nacional do Ensino Médio. Os alunos interagem na plataforma através do seu *avatar*, dentro de um enredo de Olimpíadas.

3.4 Avaliar o processo e a aprendizagem

Para avaliar o processo de aprendizagem estabelecido, estudei algumas possibilidades, tentando ver a abrangência e a qualidade do processo proposto. No sentido de clarificar a razão das minhas opções, abordarei, em seguida, a avaliação do processo de aprendizagem à luz dessas possibilidades.

3.4.1 O modelo do Pensamento Complexo

Em 1989, o departamento de educação do estado de IOWA, nos Estados Unidos, publicou um documento intitulado *A Guide to Developing Higher Order Thinking Across the Curriculum*

(Um Guia para o Desenvolvimento do Pensamento de Ordem Superior através do Currículo), abordando um modelo de pensamento integrado para se entender e trabalhar com o pensamento. Neste modelo existem quatro tipos de pensamento que interagem entre si: *Pensamento de Conteúdo*, *Pensamento Crítico*, *Pensamento Criativo* e *Pensamento Complexo* (Burklund, 1989).

O *Pensamento de Conteúdo* (ver Tabela 6) se refere à capacidade de aprender e recuperar informações e serve de base para os pensamentos crítico e complexo. Inclui o conhecimento acadêmico, o senso comum, as regras socialmente aceitas e até o conhecimento a respeito do conhecimento (Metacognição). O pensamento de conteúdo é a base sobre a qual se trabalha o pensamento crítico e complexo (Burklund, 1989; Jonassen, 2007).

Tabela 6 – Competências do Pensamento de Conteúdo (Burklund, 1989)

Competências do Pensamento de Conteúdo
Aprender
Memorizar
Entender
Recuperar
Reproduzir

O *Pensamento Crítico* (ver Tabela 7) se refere à reorganização dinâmica do conhecimento, de forma a dar-lhe significado e torná-lo utilizável. Ele organiza o conhecimento. Para o pensamento crítico são necessárias as capacidades de avaliar, analisar e relacionar. Avaliar envolve fazer juízos sobre algo, medindo relativamente a um padrão. Está relacionado a reconhecer e usar critérios em diferentes casos, reconhecer os critérios quando eles estão implícitos e ser capaz de determinar critérios apropriados. Analisar significa saber classificar, quebrando o todo em subpartes com características comuns e entender como estas subpartes funcionam e interagem a partir dessa classificação ou generalização. Relacionar envolve determinar ou impor relações no todo que está a ser analisado. Ao relacionar, comparam-se e contrastam-se coisas e ideias, procuram-se relações de causa e efeito e ligações entre os elementos. O relacionar é construído a partir do analisar, uma vez que, frequentemente, as relações são estabelecidas entre o todo e suas partes analisadas (classificadas). Envolve o pensamento lógico tanto de forma dedutiva como indutiva (Jonassen, 2007).

Tabela 7 - Competências do Pensamento Crítico (Burklund, 1989)

Competências do Pensamento Crítico
Avaliar
Avaliar informação
Determinar critérios
Estabelecer prioridades
Reconhecer falácias
Verificar
Analisar
Reconhecer padrões
Classificar
Identificar pressupostos
Identificar ideias principais
Encontrar sequências
Relacionar
Comparar/contrastar
Pensar logicamente
Inferir dedutivamente
Inferir indutivamente
Identificar relações causais

O *Pensamento Criativo* (ver Tabela 8) está relacionado às características pessoais e subjetivas para a criação de um novo conhecimento. Este conhecimento necessita de processos do pensamento crítico como analisar e avaliar e também fornece ao pensamento crítico novos conhecimentos que podem ser analisados e avaliados. Deste modo, a relação entre o pensamento crítico e criativo é dinâmica. Enquanto o pensamento crítico depende de um conhecimento mais objetivo, o pensamento criativo depende da subjetividade e de competências mais pessoais para a criação de um novo conhecimento. O pensamento criativo depende das capacidades de sintetizar, imaginar e elaborar (Jonassen, 2007). Sintetizar é a contrapartida do analisar do pensamento crítico. Sintetizar é o processo de juntar as partes de modo a criar novos conjuntos. Está relacionado com *Resumir*, *Fazer analogias*, *Planejar* e *Levantar hipóteses*. *Resumir* implica colocar juntos os elementos principais, enfatizando uma ideia e descartando a informação desnecessária. *Fazer uma analogia* implica comparar uma coisa com outra, bem diferente, mas, de certo modo, mais familiar. O pensamento analógico é diferente do *comparar e contrastar* do pensamento crítico, porque compara coisas de diferentes naturezas caracterizando

uma metáfora. *Planejar* implica juntar tarefas de modo à realização de um todo. *Levantar hipóteses* implica criar generalizações e explicações que podem ser testadas (Burklund, 1989).

Imaginar é o tipo de pensamento mais adequado ao rótulo da criatividade. Suas metas vão além dos fatos. As ideias são menos limitadas por regras lógicas e mensuráveis. A habilidade de imaginar, no modelo integrado de pensamento, vai da previsão de prováveis consequências em uma determinada situação até a especulação de remotas possibilidades reveladas pela imaginação, através de processos como a *fluência*, *visualização* e *intuição*. A fluência, *predição*, *especulação*, *visualização* e *intuição* são habilidades desejáveis ao processo de imaginar. A *fluência* está relacionada com a obtenção de ideias em grande quantidade. Um meio de conseguir a fluência é estimular as ideias mais variadas, prorrogando o processo de seleção das ideias. Na *predição* deve-se imaginar uma situação futura, dada uma situação presente, isto é, deve-se inferir uma situação em função da situação atual em um exercício de prever o futuro. Para ocorrer *especulação* deve-se pensar nas possibilidades com frases iniciadas com "e se ... ?", isto é, em condições incomuns ou inesperadas, estimulando a criatividade. Ao contrário da predição, a especulação não precisa se apoiar em lógica ou em fatos. Na *visualização* criam-se imagens mentais que, mais tarde, ajudam na comunicação. A visualização pode ser usada como um apoio mental na preparação de uma atividade ou projeto. Ajuda na observação e na memorização e pode servir de ferramenta motivacional ao "mostrar" como seu produto se parecerá no final do processo. A *intuição* refere-se a *flashes* de compreensão aparentemente instantâneos, sem a consciência de um processo racional. Um forte palpite sobre algo, sem provas concretas. Intuições são, por vezes, baseadas em conhecimentos que não se apresentam de forma consciente. As intuições servem de partida para outros tipos de conhecimentos mais objetivos. Elas podem ser rejeitadas, mas também podem ser a fagulha de ideias originais (Burklund, 1989).

Tabela 8 - Competências do Pensamento Criativo (Burklund, 1989)

Competências do Pensamento Criativo
Sintetizar
Pensar analogicamente
Resumir
Colocar hipóteses
Planificar
Imaginar
Dominar
Prever
Especular
Visualizar
Intuir
Elaborar
Expandir
Modificar
Prolongar
Trocar categorias
Concretizar
Expandir

O *Pensamento Complexo* (ver Tabela 9) reúne as capacidades dos pensamentos de conteúdo, crítico e criativo, criando processos de ordem superior, voltados para a ação. Nota-se o caráter pragmático do pensamento complexo. Para atingir um resultado, o pensamento complexo atua em etapas ou fases. Ao estimular o pensamento complexo estimulam-se os demais tipos de pensamento (ver Figura 3). Os resultados podem ser uma concepção, uma decisão ou uma solução. Para o pensamento complexo são necessárias as capacidades de resolver problemas, conceber e tomar decisões. Deste modo, ao estimular as capacidades relacionadas com o pensamento complexo se estimulam todos os demais processos de pensamento do modelo (Jonassen, 2007).

A *Resolução de Problemas* está implicada na utilização de métodos sistemáticos para esclarecer e atingir uma meta. É utilizada com problemas não rotineiros, isto é, aqueles que não se conhecem ou para os quais não existe uma fórmula ou receita para resolvê-los (Burklund, 1989). Jonassen (2007) afirma que para a resolução de problemas podem seguir-se alguns passos que mobilizam as competências:

- Apreender o problema (intuir, visualizar, dominar e identificar pressupostos);
- Investigar o problema (avaliar informação, trocar categorias, classificar, reconhecer falácias);

- Formular o problema (resumir, inferir, colocar hipóteses, concretizar, identificar ideias principais);
- Encontrar alternativas (expandir, prolongar, modificar, prever, dominar, especular);
- Escolher a solução (avaliar informação, comparar/contrastar, determinar critérios, estabelecer prioridades, verificar);
- Construir aceitação (planificar, dominar, trocar categorias, inferir, identificar relações causais, prever).

Conceber pode ser definido como inventar produtos ou informações. Os produtos podem ser uma peça de arte (pintura, composição musical, sequência de dança, instalação etc.), uma invenção mecânica ou técnica, um evento social ou cultural, um programa de computador, ou qualquer tipo de criação concreta para cumprir uma meta ou propósito (Burklund, 1989). Jonassen (2007), por sua vez, propõe alguns passos que mobilizam competências para a concepção:

- Imaginar um objetivo (dominar, trocar categorias, especular, visualizar, intuir);
- Formular um objetivo (visualizar, prever, identificar relações causais, reconhecer padrões, colocar hipóteses, planificar, raciocinar logicamente);
- Inventar um produto (dominar, planificar, expandir, concretizar, trocar categorias, pensar analogicamente, visualizar);
- Avaliar o produto (determinar critérios, avaliar informação, comparar/contrastar, reconhecer falácias, verificar);
- Rever o produto (expandir, prolongar, modificar).

Tomar Decisões significa escolher sistematicamente entre alternativas. Deve-se aprender uma alternativa sequencial e objetiva para tomada de decisões como alternativa aos processos intuitivos e também estar atento a vantagens e desvantagens de diferentes decisões técnicas. Segundo Burklund (1989), deve-se observar a ênfase objetiva e subjetiva na tomada de decisão entre os diversos domínios de ordem prática e acadêmica. Jonassen (2007) afirma que para a tomada de decisão podem considerar-se alguns passos que mobilizam as competências:

- Identificar uma questão (identificar a ideia principal, reconhecer padrões, identificar pressupostos, reconhecer falácias);
- Gerar as alternativas (dominar, prolongar, trocar categorias, colocar hipóteses, especular, visualizar);

– Avaliar as consequências (classificar, comparar/contrastar, determinar critérios, identificar relações causais, prever, pensar analogicamente);
– Tomar uma decisão (resumir, pensar logicamente, inferir, concretizar, intuir);
– Avaliar as escolhas (avaliar informação, verificar, intuir).

Para caracterizar o pensamento complexo, verificarei as competências mais gerais que são: *Resolver Problemas*; *Conceber Produtos e Informações*; *Tomar Decisões*. Ao detectar as Competências gerais do Pensamento Complexo, o modelo garante que os pensamentos de Conteúdo, Crítico e Criativo foram acionados para articular com o pensamento complexo.

Tabela 9 - Competências do Pensamento Complexo (Burklund, 1989)

Competências do Pensamento Complexo
Resolver problemas
Apreender o problema
Investigar o problema
Formular o problema
Encontrar alternativas
Escolher uma solução
Construir aceitação
Conceber Produtos e Informações
Imaginar um objetivo
Formular um objetivo
Inventar um produto
Avaliar um produto
Rever o produto
Tomar decisões
Identificar uma questão específica
Gerar alternativas
Avaliar as consequências
Fazer uma escolha
Avaliar a escolha

Figura 3 – Dependências do Pensamento Complexo (Burklund, 1989)

3.4.2 Avaliar o Pensamento Computacional

Existem várias propostas relativas a o que se deve avaliar como sendo evidências de pensamento computacional. Essas diferentes propostas se devem, em boa parte, à falta de uma definição consensual sobre o que é o pensamento computacional (Moreno-León, Robles, & Román-González, 2015). Neste trabalho focarei apenas as propostas direcionadas aos programadores em Scratch.

Karen Brennan e Mitchel Resnick (2012) sugerem que a avaliação dos produtos ou programas deve ser feita considerando três aspectos: Conceitos Computacionais, Práticas Computacionais e Perspectivas Computacionais.

Conceitos Computacionais se referem aos elementos da linguagem de programação que são Sequências, Loops, Paralelismo, Eventos, Condicionais, Operadores e Dados. A avaliação é baseada em portefólios de projetos. Brennan e Resnick utilizam uma ferramenta chamada *User Analysis* do conjunto de ferramentas Scrape (http://happyanalyzing.com), desenvolvida no College of New Jersey, que representa de forma gráfica os comandos Scratch do portefólio de um programador, através da sua conta na comunidade *online* do Scratch. O resultado é uma imagem gráfica que permite avaliar a proficiência do programador. Confira as imagens da Figura 4 e Figura 5.

As *Práticas Computacionais* estão relacionadas com a maneira como os programadores procedem enquanto desenvolvem o programa e abrange Experimentação e interação, Teste e depuração, Reutilização e recombinação, Abstração e modelagem. A avaliação é feita através de entrevista com os alunos.

As *Perspectivas Computacionais* se referem a uma visão mais panorâmica do programador e suas relações com o meio. Se referem a Expressão (O que eu posso criar com o Scratch?), Conexão (Posso fazer coisas diferentes quando me interajo com os outros?) e Questionamento (Posso usar o Scratch para questionar e dar sentido ao mundo?). A avaliação é feita através de desenhos de cenário, isto é, colocando o aluno para realizar atividades em pequenos projetos Scratch que devem ser depurados e incrementados.

Linda Seiter e Brendan Foreman (2013) propõem um modelo de Progression of Early Computational Thinking Model (PECM). Trata-se de um modelo de avaliação do pensamento computacional a partir dos conceitos “Procedimentos e algoritmos”, “Decomposição de Problemas”, “Paralelização e Sincronização”, “Abstração e Representação de dados”. Para

chegar a estes conceitos os autores mapeiam os padrões do desenho que o programador Scratch utiliza ao programar: Animação de Aparência, Animação por Movimentação, Conversas entre personagens, Colisões, Pontuação de jogos, Interação com o usuário. Estes padrões de desenho são obtidos a partir das evidências de comandos utilizados na linguagem Scratch que são os comandos de Aparência, Som, Movimento, Variáveis, Sequência & Looping, Expressões booleanas, Operadores, Condicionais, Controles, Eventos da interface com o usuário, Paralelização, Inicializar Localização e Inicializar Aparência.

As propostas de Brennan e Resnick e de Seiter e Foreman se baseiam em uma análise manual para avaliar o pensamento computacional, assim como a maioria das abordagens de avaliação do pensamento computacional codificado em linguagem Scratch (Moreno-León et al., 2015). A análise manual se faz necessária na proposta de Brennan e Resnick para avaliar os desenhos de cenários para se obter as *Perspectivas Computacionais* e na proposta de Seiter e Foreman para se obter os padrões do desenho.

Outra maneira de avaliar o pensamento computacional é através da ferramenta Dr. Scratch que detalharei a seguir. Esta ferramenta atribui uma pontuação a um programa que reflete o nível de desenvolvimento do pensamento computacional. Esta abordagem não utiliza entrevistas como a proposta de Brennan e Resnick. Dr. Scratch se baseia na análise do código estático como a proposta de Seiter e Foreman. No entanto, Dr. Scratch automatiza a análise, sem necessidade de uma análise manual.

A ferramenta Dr. Scratch fornece um valor de 0 a 21 e pode ser aplicada a um único programa Scratch para uma avaliação somatória de um projeto ao final de uma oficina ou disciplina curricular, enquanto a proposta de Brennan e Resnick analisa um portefólio com programas acumulados de longa data. Pelas facilidades aqui descritas, utilizarei nesta investigação a ferramenta Dr. Scratch para avaliar a amostra dos programas desenvolvidos pelos alunos.

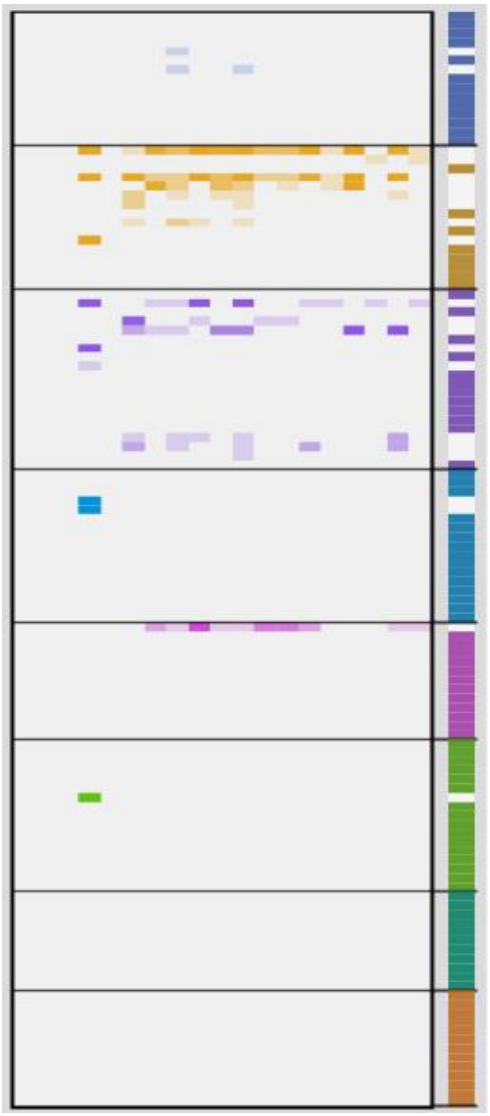


Figura 4 - Imagem Scrape para Programador Inexperiente

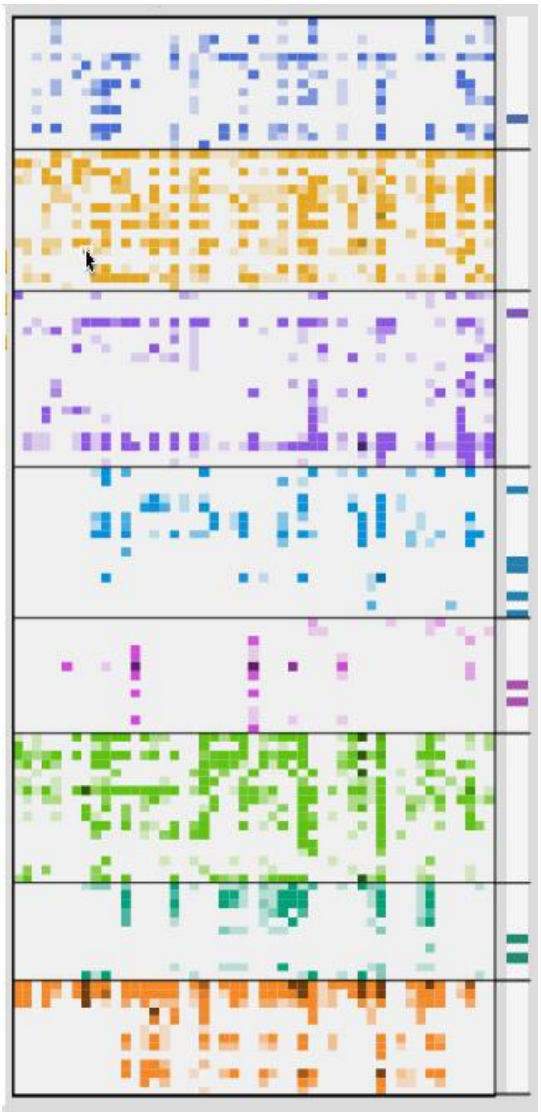


Figura 5 - Imagem Scrape para Programador Experiente

3.4.2.1 A ferramenta Dr. Scratch

Dr. Scratch é uma ferramenta desenvolvida na universidade Rey Juan Carlos na Espanha. Trata-se de uma ferramenta para analisar programas Scratch inferindo sobre o pensamento computacional do seu desenvolvedor a partir de uma análise estática do programa em linguagem Scratch. Dr. Scratch utiliza plug-ins desenvolvidos para a ferramenta Hairball (Moreno-León et al., 2015) que, por sua vez, foi inspirada na lint, uma ferramenta do Unix desenvolvida em 1979 para avaliar código estático em linguagem C. Hairball e Dr. Scratch se propõem a fornecer uma avaliação sumativa e a possibilitar uma avaliação formativa, indicando possíveis melhorias no código Scratch (Boe et al., 2013; Moreno-León et al., 2015).

3.4.2.2 Indicadores de Pensamento Computacional

Dr. Scratch estabelece Abstração, Paralelismo, Lógica, Sincronização, Controle de fluxo, Interatividade com o usuário e a Representação de dados, como aspectos a serem referenciados para estimar o pensamento computacional evidenciado pelos programadores. A cada um desses aspectos é atribuída uma pontuação de 0 a 3, a partir dos comandos utilizados no código estático em linguagem Scratch. Os sete aspectos cujos valores variam de 0 a 3, estabelecem um indicador que varia de 0 a 21. O mapeamento dos comandos e os aspectos analisados é feito automaticamente pela ferramenta, seguindo a Tabela 10. A ferramenta é de fácil utilização e pode ser acessada em http://drscratch.programamos.es.

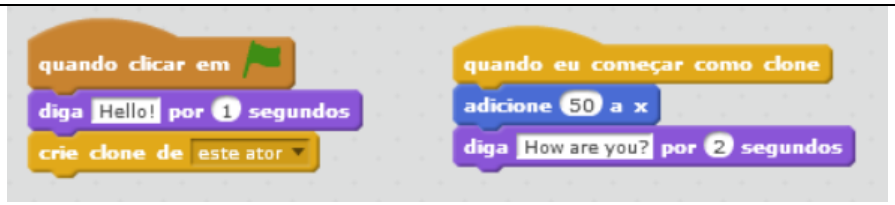
Tabela 10 - Pontuação do Pensamento Computacional

Conceito	Básico (1)	Em desenvolvimento (2)	Proficiência (3)
Abstração	Mais de um *script* e mais de um objeto	Definição de Blocos	A utilização de clones
Paralelismo	Dois *scripts* em bandeira verde	Dois *scripts* quando uma tecla for pressionada ou dois scripts para quando o objeto for Clicado	Dois ou mais scripts quando: recebe uma mensagem, ou cria clone, ou quando interface (cronômetro, áudio, vídeo) maior que um parâmetro, ou quando muda o pano de fundo.
Lógica	Uso do condicional SE	Uso do condicional SE-SENÃO	Operações lógicas (E, OU, NÃO)
Sincronização	Usando ESPERE	Envia (sem espera) e recebe mensagem, ou parar tudo, ou usar l comando pare.	Usa o comando aguarda até que, quando a muda pano de fundo, envia uma mensagem e espera
Controle de fluxo	Sequência de blocos	Uso dos loops "Repita" ou "Sempre"	Uso do loop "Repita até que"
A interatividade com o usuário	Bandeira verde	Verifica tecla pressionada, ou objeto clicado, ou pedir resposta e esperar a entrada do teclado, ou verifica posição ou clique do mouse.	Verifica as interfaces de cronômetro, áudio, ou vídeo comparando como um parâmetro.
Representação de dados	Modificação nas propriedades dos objetos.	Usa variáveis	Usa listas.

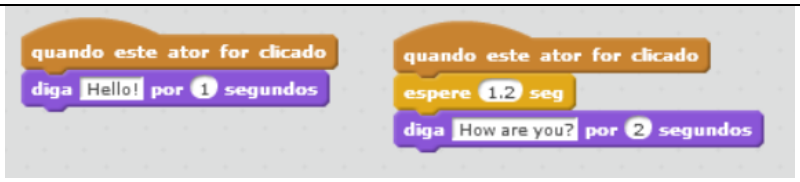
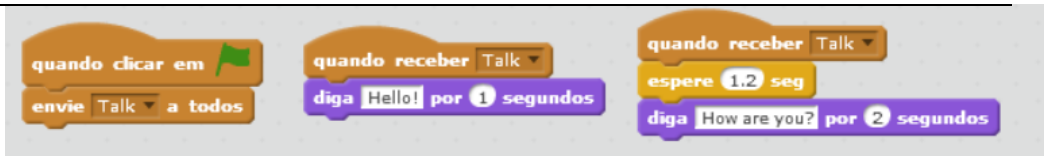
Para melhor entendimento do mapeamento representado na Tabela 10, descrevemos alguns exemplos de programas cuja pontuação obtida pela aplicação da ferramenta Dr. Scratch

é distinta. Os exemplos originais (programas em Scratch) podem ser acessados em: http://goo.gl/U3u2v5 .

Abstração

Nível	Exemplo
1	
2	
3	

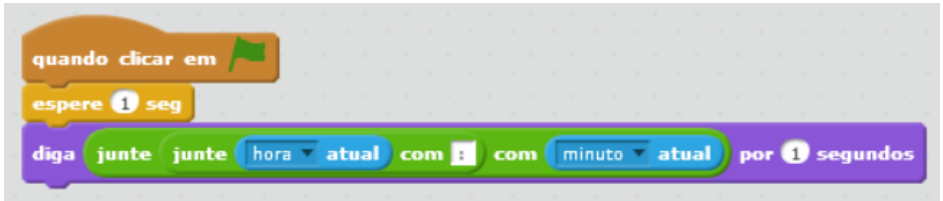
Paralelismo

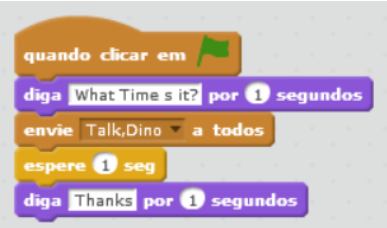
Nível	Exemplo
1	
2	
3	

Lógica

Nível	Exemplo
1	
2	
3	

Sincronização

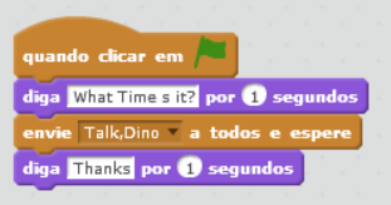
Nível	Exemplo
1	Ator Gato:  Ator Dinossauro: 
2	Ator Gato:



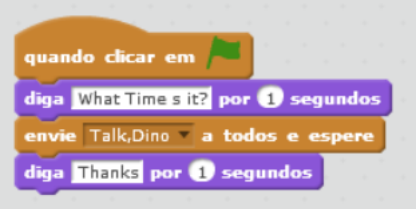
Ator Dinossauro:

3

Ator Gato:

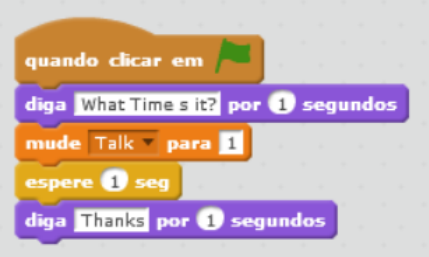


Ator Dinossauro:



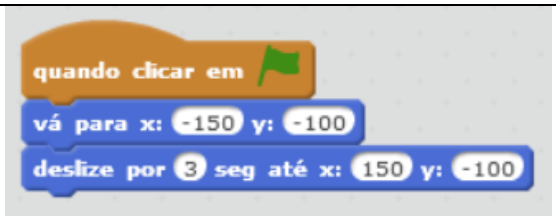
3

Ator Gato:

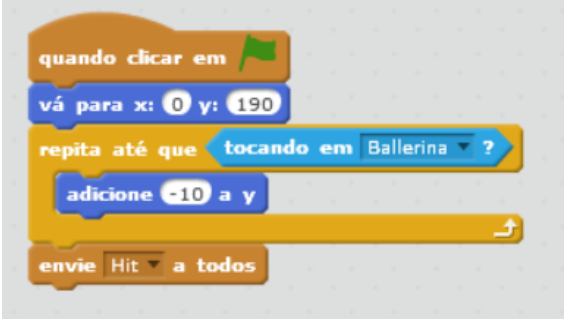


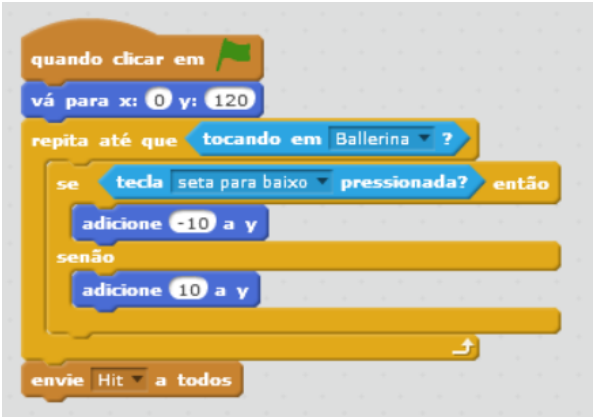
Ator Dinossauro:

Controle de fluxo

Nível	Exemplo
1	
2	
3	

Interatividade com o usuário

Nível	Exemplo
1	

2	
3	

Representação de dados

Nível	Exemplo
1	
2	

3

3.4.3 A taxonomia de Bloom

No final dos anos 40 do século XX, Benjamin Bloom teve a iniciativa de criar um grupo de discussão para gerar um modelo que facilitasse a elaboração de testes por diversos professores universitários seguindo o mesmo critério. Com o apoio da Associação Americana de Psicologia, em 1956 foi publicada a obra *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain* (Bloom, Engelhart, Furst, Hill, & Krathwohl, 1956). Este trabalho mapeava, de forma hierárquica, os níveis de pensamento do mais simples ao mais complexo. O modelo gerado ficou conhecido como taxonomia de Bloom (Krathwohl, 2002). Esta taxonomia compreende seis níveis cognitivos: conhecimento, compreensão, aplicação, análise, síntese e avaliação. As características de cada nível estão descritas na Tabela 11 (Prata & Nascimento, 2007).

Na década de 1990, também um grupo se reúne para repensar a taxonomia de Bloom de 1956. Neste grupo destaca-se David Krathwohl, que participou da elaboração da taxonomia original, e Lorin Anderson, um ex-aluno de Benjamin Bloom. O resultado do trabalho deste grupo, foi publicado no livro *A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom's Taxonomy for Educational Objectives* (Anderson & Krathwohl, 2001).

Tabela 11 – Taxonomia de Bloom, proposta em 1956 (Prata & Nascimento, 2007)

Nível	Descrição	Verbos Associados
1. Conhecimento	Lembrar informações sobre: fatos, datas, palavras, teorias, métodos, classificações, lugares, regras, critérios, procedimentos etc.	Enumerar, definir, descrever, identificar, denominar, listar, nomear, combinar, realçar, apontar, relembrar, recordar, relacionar, reproduzir, solucionar, declarar, distinguir, rotular, memorizar, ordenar e reconhecer.
2. Compreensão	Entender a informação ou o fato, captar seu significado, utilizá-la em contextos diferentes.	Alterar, construir, converter, decodificar, defender, definir, descrever, distinguir, discriminar, estimar, explicar, generalizar, dar exemplos, ilustrar, inferir, reformular, prever, reescrever, resolver, resumir, classificar, discutir, identificar, interpretar, reconhecer, redefinir, selecionar, situar e traduzir.
3. Aplicação	Aplicar o conhecimento em situações concretas.	Aplicar, alterar, programar, demonstrar, desenvolver, descobrir, dramatizar, empregar, ilustrar, interpretar, manipular, modificar, operacionalizar, organizar, prever, preparar, produzir, relatar, resolver, transferir, usar, construir, esboçar, escolher, escrever, operar e praticar.
4. Análise	Identificar as partes e suas inter-relações.	Analisar, reduzir, classificar, comparar, contrastar, determinar, deduzir, diagramar, distinguir, diferenciar, identificar, ilustrar, apontar, inferir, relacionar, selecionar, separar, subdividir, calcular, discriminar, examinar, experimentar, testar, esquematizar e questionar.
5. Síntese	Combinar partes não organizadas para formar um todo.	Categorizar, combinar, compilar, compor, conceber, construir, criar, desenhar, elaborar, estabelecer, explicar, formular, generalizar, inventar, modificar, organizar, originar, planejar, propor, reorganizar, relacionar, revisar, reescrever, resumir, sistematizar, escrever, desenvolver, estruturar, montar e projetar.
6. Avaliação	Julgar o valor do conhecimento.	Avaliar, averiguar, escolher, comparar, concluir, contrastar, criticar, decidir, defender, discriminar, explicar, interpretar, justificar, relatar, resolver, resumir, apoiar, validar, escrever uma revisão sobre, detectar, estimar, julgar e selecionar.

Na versão publicada na década de 1950, a taxonomia compreendia a dimensão cognitiva, afetiva e psicomotora, sendo que a dimensão cognitiva, aquela que nos interessa abordar aqui, incluía seis categorias principais, cada uma descrita por um substantivo. A versão revista por Anderson e colegas mantém a mesma organização em seis categorias mas estas são distribuídas de uma forma diferente e identificadas por verbos em vez de nomes. A categoria síntese, que era a penúltima, passou a ser a última com a designação de criar. As novas

categorias foram, assim, descritas conforme se pode observar na Tabela 12 (Ferraz & Belhot, 2010).

Tabela 12 - Processos Cognitivos da Taxonomia de Bloom 2001 (Ferraz & Belhot, 2010)

Nível	Descrição	Verbos Associados
1. Lembrar	Relacionado a reconhecer e reproduzir ideias e conteúdos. Reconhecer requer distinguir e selecionar uma determinada informação e reproduzir ou recordar está mais relacionado à busca por uma informação relevante memorizada.	Reconhecendo e Reproduzindo.
2. Entender	Relacionado a estabelecer uma conexão entre o novo e o conhecimento previamente adquirido. A informação é entendida quando o aprendiz consegue reproduzi-la com suas "próprias palavras".	Interpretando, Exemplificando, Classificando, Resumindo, Inferindo, Comparando e Explicando.
3. Aplicar	Relacionado a executar ou usar um procedimento numa situação específica e pode também abordar a aplicação de um conhecimento numa situação nova	Executando e Implementando.
4. Analisar	Relacionado a dividir a informação em partes relevantes e irrelevantes, importantes e menos importantes e entender a inter-relação existente entre as partes.	Diferenciando, Organizando, Atribuindo e Concluindo.
5. Avaliar	Relacionado a realizar julgamentos baseados em critérios e padrões qualitativos e quantitativos ou de eficiência e eficácia.	Checando e Criticando.
6. Criar	Significa colocar elementos junto com o objetivo de criar uma nova visão, uma nova solução, estrutura ou modelo utilizando conhecimentos e habilidades previamente adquiridos. Envolve o desenvolvimento de ideias novas e originais, produtos e métodos por meio da percepção da interdisciplinaridade e da interdependência de conceitos.	Generalizando, Planejando e Produzindo.

Por vezes ficamos em dúvida para classificar um determinado processo, a partir de uma narrativa. O uso das subclasses dos processos cognitivos, além de detalhar as características do processo, aumenta o vocabulário de verbos, facilitando a reconhecimento dos processos analisados segundo o modelo da taxonomia. As subclasses dos processos cognitivos podem ser vistas nas tabelas de Tabela 13 até Tabela 18.

Tabela 13 - Processo Cognitivo Lembrar com base em Mayer (2002) e Ferraz e Belhot (2010)

Processo Cognitivo (Classe e subclasse)	Descrição e Verbos Alternativos
Lembrar	Relacionado a reconhecer e reproduzir ideias e conteúdos. Reconhecer requer distinguir e selecionar uma determinada informação e reproduzir ou recordar está mais relacionado à busca por uma informação relevante memorizada.
Reconhecendo	Envolve localizar o conhecimento na memória de longo prazo que é consistente com o material apresentado. Verbo alternativo: Identificando.
Recordando	Envolve recuperar conhecimentos importantes da memória de longo prazo. Verbo alternativo: Recuperando.

Tabela 14 – Processo Cognitivo de Entender com base em Mayer (2002) e Ferraz e Belhot (2010)

Processo Cognitivo (Classe e subclasse)	Descrição e Verbos Alternativos
Entender	Relacionado a estabelecer uma conexão entre o novo e o conhecimento previamente adquirido. A informação é entendida quando o aprendiz consegue reproduzi-la com suas "próprias palavras".
Interpretando	Ocorre quando um aluno é capaz de converter informação de uma forma de representação para outro. Verbos alternativos: Esclarecendo, Parafraseando, Representando, Traduzindo.
Exemplificando	Ocorre quando um estudante encontra um exemplo ou instância do conceito geral ou princípio específico. Verbos alternativos: Ilustrando, Instanciando.
Classificando	Ocorre quando um aluno determina que alguma coisa (por exemplo, uma instância ou um exemplo em particular) pertence a uma certa categoria (por exemplo, conceito ou princípio). Verbos alternativos: Categorizando, Subordinando.
Resumindo	Ocorre quando um aluno produz uma breve declaração que representa informações apresentadas ou resumos de um tema geral. O comprimento do resumo, em certa medida depende do comprimento do material apresentado. Verbos alternativos: Abstraindo, Generalizando.
Inferindo	Envolve desenhar uma informação conclusão lógica a partir de uma informação apresentada. Verbos alternativos: Concluindo, Extrapolando, Interpolando, Prevendo.
Comparando	Envolve a detecção de semelhanças e diferenças entre dois ou mais objetos, eventos, ideias, problemas ou situações.

	Verbos alternativos: Contrastando, Mapeando, Correspondendo
Explicando	Ocorre Quando um aluno constrói mentalmente um modelo de causa e efeito, de um sistema ou sequência, e o utiliza. Verbo alternativo: Construindo Modelos

Tabela 15 – Processo Cognitivo Aplicar com base em Mayer (2002) e Ferraz e Belhot (2010)

Processo Cognitivo (Classe e subclasse)	Descrição e Verbos Alternativos
Aplicar	Relacionado a executar ou usar um procedimento numa situação específica e pode também abordar a aplicação de um conhecimento numa situação nova
Executando	Ocorre Quando o aluno aplica um procedimento para uma tarefa familiar. Verbo alternativo: Realizando
Implementando	Ocorre quando um aluno aplica um ou mais procedimentos para uma tarefa desconhecida. Note-se que nesta tarefa de avaliação, os estudantes não devem apenas aplicar um procedimento (ou seja, envolver-se em execução), mas também contar com a compreensão conceitual do problema e procedimento. Assim, ao contrário de execução, que depende quase que exclusivamente dos Processos cognitivos associados com ao Aplicar, de execução envolve processos cognitivos associados com o Entender e Aplicar. Verbo alternativo: Usando

Tabela 16 - Processo Cognitivo Analisar com base em Mayer (2002) e Ferraz e Belhot (2010)

Processo Cognitivo (Classe e subclasse)	Descrição e Verbos Alternativos
Analisar	Relacionado a dividir a informação em partes relevantes e irrelevantes, importantes e menos importantes e entender a inter-relação existente entre as partes.
Diferenciando	Ocorre quando um estudante discrimina entre partes relevantes e partes irrelevantes ou partes importantes de não importantes de um material apresentado. Verbos alternativos: Discriminando, Selecionando, Distinguindo, Focando.
Organizando	Consiste em determinar como os elementos se encaixam ou funcionam dentro de uma estrutura. Verbos alternativos: Encontrando Coerência, Integrando, Delineando, Repassando, ou Estruturando.
Atribuindo	Ocorre quando um estudante é capaz de determinar o ponto de vista, viés, valores, ou a intenção subjacente material apresentado. Verbo alternativo: Desconstruindo.

Tabela 17 - Processo Cognitivo Avaliar com base em Mayer (2002) e Ferraz e Belhot (2010)

Processo Cognitivo (Classe e subclasse)	Descrição e Verbos Alternativos
Avaliar	Relacionado a realizar julgamentos baseados em critérios e padrões qualitativos e quantitativos ou de eficiência e eficácia.
Verificação	Ocorre quando um aluno detecta inconsistências ou falácias dentro de um processo ou produto, determinando se um processo ou produto tem consistência interna, ou detecta a eficácia de um procedimento que está sendo implementado. Quando combinado com Planejando (um processo cognitivo na categoria, Criar) e com Implementando (um processo cognitivo na categoria Aplicar) Verificando envolve determinar o quão bem o plano está funcionando. Verbos alternativos: Coordenando, Detectando, Monitorando, Testando.
Criticando	Ocorre quando um estudante detecta inconsistências entre um produto ou operação e alguns critérios externos, determinando se um produto tem consistência externa, ou julgando a adequação de um procedimento de problema dado. Criticando está no cerne do que tem sido chamado de pensamento crítico. Ao criticar, os alunos julgam os méritos de um produto ou operação com base em critérios especificados ou e padrões determinados pelo estudante. Verbos alternativos: Julgando

Tabela 18 - Processo Cognitivo Criar com base em Mayer (2002) e Ferraz e Belhot (2010)

Processo Cognitivo (Classe e subclasse)	Descrição e Verbos Alternativos
Criar	Significa colocar elementos junto com o objetivo de criar uma nova visão, uma nova solução, estrutura ou modelo utilizando conhecimentos e habilidades previamente adquiridos. Envolve o desenvolvimento de ideias novas e originais, produtos e métodos por meio da percepção da interdisciplinaridade e da interdependência de conceitos.
Gerando	Envolve inventar hipóteses alternativas com base em critérios. Quando Gerando transcende os limites ou restrições de conhecimento prévio e as teorias existentes, envolve o pensamento divergente e constitui o núcleo do que pode ser chamado de pensamento criativo. Em Gerando, é dada a um estudante uma descrição de um problema e ele deve produzir soluções alternativas. Para cada uma dessas avaliações, são necessários critérios de pontuação explícitos. Verbos alternativos: Levantar Hipóteses
Planejando	Envolve a elaboração de um método para realizar uma tarefa. No entanto, o planejamento não chega a levar a cabo os passos para criar a atual solução dada para um problema. No planejamento, um estudante pode estabelecer submetas (ou seja, quebrar uma tarefa em subtarefas a serem executadas quando estiver resolvendo o problema). Os professores geralmente pulam o planejamento como objetivo e, ao invés disso, indicam os seus objetivos em

	termos de produto, que é o estágio final do processo criativo. Quando isso acontece, assume-se que o planejando está implícito no objetivo de Produzindo. Neste caso, o planejamento é susceptível de ser realizada pelo aluno secretamente, no curso da construção de um produto (ou seja, Produzindo). No planejamento, um aluno desenvolve um método para solução de um problema dado. Verbos alternativos: Desenhando
Produzindo	Envolve inventar um produto. Na produção, a um estudante é dada uma descrição funcional de um objetivo e o aluno deve criar um produto que satisfaça a descrição. Verbos alternativos: Construindo

A equipe de revisão da taxonomia de Bloom notou que, se os objetivos listados por professores são da forma verbo-nome, por exemplo, os alunos serão capazes de lembrar o comando sorteia, que sorteia um número dentro de um limite estabelecido. O verbo é *lembrar* e o nome é o *comando sorteia*, sendo que o nome indica o que vai ser ensinado, ou seja, o *conteúdo* e o verbo indica como será ensinado, ou seja, *o processo*. A partir dessa observação, além da dimensão de Processos Cognitivos, criou-se uma nova dimensão chamada de Conhecimento. De forma a classificar o conteúdo a ser ensinado, conforme mostra a Tabela 19 (Ferraz & Belhot, 2010; Krathwohl, 2002).

Tabela 19 - Dimensão Conhecimento na taxonomia de Bloom revisada (Ferraz & Belhot, 2010)

Conhecimento	Descrição
Factual	Conteúdo básico que o aluno deve ter para se familiarizar e trabalhar na disciplina sem esforço para o seu entendimento. – Conhecimento da Terminologia; – Conhecimento de detalhes e elementos específicos.
Conceitual	Entendimento das inter-relações entre os elementos básicos em uma estrutura maior que os fazem funcionar em conjunto. O entendimento é importante, mas sua aplicação não é exigida. – Conhecimento de classificações e categorias; – Conhecimento de princípios e generalizações; – Conhecimento de teorias, modelos e estruturas.
Processual	Conhecimento de "como fazer", utilizando habilidades, algoritmos, técnicas e métodos adequados. O conhecimento abstrato começa a ser estimulado dentro de um contexto, mas ainda não de forma interdisciplinar. – Conhecimento de habilidades e algoritmos específicos; – Conhecimento técnicas e métodos específicos; – Conhecimento de critérios para uso dos procedimentos específicos.
Metacognitivo	Conhecimento da cognição em geral com consciência da própria cognição adquirida de um conteúdo específico. Espera-se o uso

	conhecimento adquiridos de forma interdisciplinar para resolução de problemas e escolhendo teorias e métodos criteriosamente. – Conhecimento estratégico; – Conhecimento sobre a cognição em geral, incluindo a consciência e conhecimento de sua própria cognição; – Autoconhecimento.

3.4.3.1 Críticas à taxonomia de Bloom

As taxonomias têm sido utilizadas para atingir objetivos dos cursos, desenvolver métodos de ensino, como técnicas de avaliação, como veículo para os educadores avaliarem os seus próprios métodos de ensino, para reforma curricular e para ensinar como se tornar melhores professores. (Spivey, 2007). Para contextualizar a Taxonomia de Bloom, a versão revisada comenta quatro questões organizadoras que colocam o uso da taxonomia em perspectiva (Anderson & Krathwohl, 2001; Spivey, 2007). Veja Tabela 20.

Tabela 20 - Questões Organizadoras (Spivey, 2007)

Questão	Descrição
Questão de Aprendizagem	O que é importante para os alunos aprenderem na escola e no limitado tempo disponível de sala de aula?
Questão da Instrução	Como é que um plano e a instrução entregues irão resultar em altos níveis de aprendizagem para um grande número de alunos?
Questão de Avaliação	Como é que os instrumentos de avaliação e os procedimentos planejados fornecem informações precisas sobre quão bem os alunos estão aprendendo?
Questão de Alinhamento	Como é que se assegura que os objetivos, instrução e avaliação são consistentes entre si?

Andreia de Jesus e Gláucia Brito (Jesus & Brito, 2009), discorrendo sobre o ensino de programação, constatam que os professores não estão preparados para ensinar resolução de problemas. As aulas costumam ser baseadas em modelos prontos, com soluções do próprio professor, levando o aluno a reproduzir técnicas e estratégias memorizadas, sem uma aprendizagem mais profunda. Jesus e Brito comentam, também, que nas aulas os alunos mais preparados não são desafiados, enquanto os menos aptos não recebem a atenção devida que os levem a superar os desafios da programação. Por estes e outros fatores, Jesus e Brito preocupam-se com a grande evasão das disciplinas de ensino de programação e algoritmos e consequentemente dos cursos de computação.

Em outro trabalho, Elieser de Jesus e André Raabe (Jesus & Raabe, 2009) sugerem uma adaptação da taxonomia de Bloom existente para o curso de ensino de programação e algoritmos para o curso de computação, como mostra a Tabela 21. Esta tabela foi criada a partir do ensino de programação para o curso de computação em linguagem como C ou Pascal. Mas, no contexto do ensino de programação de computadores para formação de professores, utilizando a linguagem Scratch, esta tabela não se aplica. Objetivos como "modificar o código de um *loop* do tipo *for* para um do tipo *while*", "diferenciar um método *construtor*" ou se preocupar com "critérios de eficiência do código", não se mostram adequados. Aspectos como o paralelismo de código e a clonagem de objetos, característicos da linguagem Scratch, não aparecem nesta tabela. Sendo assim, uma outra adaptação deve ser feita para utilizar a taxonomia de Bloom no contexto deste trabalho. No entanto a classe Criar, da adaptação apresentada por Jesus e Raabe, me parece adequada. O objetivo "Aplicar algoritmos conhecidos em uma combinação não familiar para o aluno", se aplica perfeitamente aos projetos finais que solicito aos alunos.

Tabela 21 - Adaptação da Taxonomia de Bloom ao Ensino de Programação (Jesus, 2009)

Categoria	Interpretação em programação
Lembrar	Recuperação do conhecimento relevante da memória de longo termo *Identificar elementos específicos em um trecho de código.* *Reconhecer a implementação de um determinado conceito.* *Reconhecer a descrição mais apropriada para um determinado conceito. Lembrar de um conceito, processo, algoritmo, etc.* *Listar operadores de acordo com a ordem de precedência.* Definir o propósito de um método construtor. Descrever um determinado padrão de projeto. Citar os nomes dos tipos de *loops* em uma linguagem de programação. Listar *N* métodos
Entender	Construção de significados através de diferentes tipos de linguagens. *Escrever em pseudo-código, fluxograma ou linguagem natural um programa que calcule uma fórmula bem conhecida.* *Completar partes em falta de um programa utilizando fragmentos de código.* *Explicar com palavras o comportamento de um trecho de código.* Predizer valores de variáveis depois da execução de um trecho de código. Traduzir um algoritmo de uma forma de representação para outra. Explicar um conceito, algoritmo ou padrão de projeto. Apresentar exemplos de um conceito, algoritmo ou padrão de projeto.

Categoria	Interpretação em programação
Aplicar	Utilização de processos conhecidos para executar ou implementar. *Implementar um programa utilizando como exemplo um código que resolva um problema semelhante.* *Implementar ordenação de vetores não numéricos com alunos que já tenham ordenado vetores numéricos.* *Executar mentalmente expressões seguindo as regras de precedência.* Resolver um problema familiar, mas com dados ou ferramentas não familiares. Modificar o código de um *loop* do tipo *for* para um do tipo *while.*
Analisar	Decomposição de um problema em suas partes constituintes e determinação das relações entre as partes e o todo. *Dividir uma tarefa de programação em suas partes componentes.* *Organizar as partes componentes para atingir um objetivo geral.* *Identificar componentes críticos para o desenvolvimento.* Identificar componentes ou requisitos não importantes. Diferenciar um método construtor dos demais métodos de uma classe.
Avaliar	Realização de julgamentos baseados em critérios e padrões. *Determinar se um código satisfaz os requisitos definindo uma estratégia de teste apropriada.* *Criticar a qualidade de um código baseando-se em boas práticas de programação ou critérios de eficiência do código.* *Avaliar qual de dois algoritmos que resolvem a mesma tarefa é mais adequado.*
Criar	Juntar elementos para formar um todo coerente e funcional. *Propor algoritmo, processo ou estratégia alternativa para um problema. Hipotetizar que uma nova combinação de algoritmos resolverá o problema. Construir um programa utilizando algoritmos inventados.* *Aplicar algoritmos conhecidos em uma combinação não familiar para o aluno.*

Relacionando a taxonomia de Bloom com o ensino de programação de computadores Raymond Lister afirma:

> Os dois níveis inferiores enfatizam a habilidade de ler e compreender o código, os dois níveis intermediários enfatizam a escrita de pequenos fragmentos de código, mas dentro de um contexto bem definido, e os dois níveis superiores enfatizam a escrita do programa não-trivial completa. Os alunos devem primeiro ser ensinados a ler programas antes de escrever programas. (Lister, 2000, p. 159)

3.4.3.2 A taxonomia SOLO

Entre as críticas que são feitas a taxonomia de Bloom, as mais recorrentes incidem na falta de consenso na classificação das atividades nas classes estabelecidas e na avaliação de que a hierarquia pretendida é falha (Biggs, 2003; A. Jesus & Brito, 2009; Whalley et al., 2006).

Quanto à dificuldade de classificação Jacqueline Whalley e outros comentam, depois de utilizar a taxonomia de Bloom:

> Em primeiro lugar, categorizar Testes de Múltipla Escolha por complexidade cognitiva, aplicando a taxonomia de Bloom, tem-se revelado um desafio difícil, até mesmo para um grupo experiente de educadores de programação. Isto pode sugerir algumas deficiências na taxonomia de Bloom quando aplicada a problemas de programação, ou ser uma manifestação do nível atual de compreensão da aplicação da taxonomia. Isso também indica que a avaliação de programação de forma justa e consistente é uma tarefa complexa e desafiadora, carente de estruturas e ferramentas que possam auxiliar os educadores neste propósito. (Whalley et al., 2006, p. 251)

Quanto à hierarquia da classe Ana Ferraz e Renato Belhot destacam que:

> Embora a nova taxonomia mantenha o *design* hierárquico da original, ela é flexível, pois possibilitou considerar a possibilidade de interpolação das categorias do processo cognitivo quando necessário, devido ao fato de que determinados conteúdos podem ser mais fáceis de serem assimilados a partir do estímulo pertencente a uma mais complexa. Por exemplo, pode ser mais fácil entender um assunto após aplicá-lo e só então ser capaz de explicá-lo. (Ferraz & Belhot, 2010, p. 7)

Um estudo foi realizado, na Universidade de Kent, na Inglaterra, contemplando 54 avaliações de alunos do primeiro ano, estudantes de ciência da computação, durante um ano. Foram examinados por um painel de cinco acadêmicos do departamento, que foram convidados a decidir quais os níveis da taxonomia de Bloom estavam sendo avaliados. Os resultados são apresentados na Tabela 22. (Johnson & Fuller, 2006, p. 121).

Tabela 22 - Classificação de Testes do Estudo de Johnson

Bloom Level	Assessor 1	Assessor 2	Assessor 3	Assessor 4	Assessor 5	Mean
Knowledge	54	4	54	43	53	42
Comprehension	54	13	54	9	52	37
Application	51	43	54	5	29	36
Analysis	25	17	9	0	3	11
Synthesis	0	2	6	1	2	2
Evaluation	0	3	2	0	0	1

A partir da tabela do estudo de Johnson e Fuller, duas observações podem ser feitas. A primeira é que, ratificando o que afirmou Whalley, é difícil chegar a um consenso sobre como usar a taxonomia de Bloom para classificação dos testes. A segunda observação é que, embora não haja um consenso absoluto, nota-se uma tendência de se avaliar os primeiros três níveis da taxonomia, deixando os três últimos níveis, referentes ao pensamento de alto nível, em segundo

plano. Por outras palavras, o estudo mostrou que nessa amostra das avaliações na Universidade de Kent, 89% se referem aos níveis baixos da taxonomia de Bloom, enquanto apenas 11% se referem aos níveis mais altos.

Tabela 23 - Níveis da Taxonomia SOLO (Biggs, 2003; Biggs, 2011)

Níveis de Entendimento	Descrição	Verbos Associados
Pré-estrutural	O aluno não responde a contento, indicando falha no aprendizado.	
Uni-estrutural	O aluno faz conexões simples e consegue lidar com fatos básicos, mas mostra pouca evidência que entende o significado dos fatos.	Memorizar, identificar, reconhecer, contar, definir, desenhar, encontrar, rotular, corresponder, nomear, cotar, lembrar, recitar, ordenar, dizer, escrever, imitar.
Multiestrutural	O aluno conhece um certo número de tópicos e pode fazer conexões entre eles, mas não percebe o significado do todo.	Classificar, descrever, listar, relatar, discutir, ilustrar, selecionar, narrar, computar, sequenciar, esboçar, separar
Relacional	O aluno sabe o significado das partes e relaciona-as com o tudo.	Aplicar, integrar, analisar, explicar, prever, concluir, resumir, rever, arguir, transferir, planejar, caracterizar, comparar, contrastar, diferenciar, organizar, debater, compor um caso, construir, rever e reescrever, examinar, traduzir, parafrasear, resolver problemas.
Abstrato Estendido	O aluno faz conexões internas e além do que foi ensinado e é capaz de aplicar os princípios e ideias em uma nova situação.	Teorizar, Levantar hipóteses, generalizar, refletir, gerar, criar, compor, inventar, causar, provar a partir de princípios básicos, fabricar um caso original, resolver a partir de princípios básicos.

A taxonomia SOLO - Structure of Observed Learning Outcomes, aparece com a promessa de resolver estas questões e simplificar a avaliação com uma taxonomia unidimensional com apenas quatro níveis relevantes. A taxonomia SOLO foi desenvolvida por John B. Biggs e Kevin F. Collis para avaliar os artefatos produzidos pelos alunos no seu processo de aprendizado. Verificando assim o que o aluno é capaz de produzir, sem tentar imaginar sua estrutura cognitiva, baseando-se apenas nos resultados observáveis. A Estrutura de Respostas Observáveis de Aprendizado está descrita na Tabela 23 (Biggs, 2003; Biggs, 2011).

Pelo exposto, nota-se que a taxonomia de SOLO dá ênfase ao produto entregue pelo aluno, enquanto a taxonomia de Bloom enfatiza o processo de ensino-aprendizagem. Para este trabalho, que pretende apontar contribuições para o ensino de programação, achei mais

conveniente o uso da taxonomia de Bloom, focando os objetivos cognitivos das tarefas propostas na disciplina.

3.4.3.3 Bloom e Pensamento Computacional

Em um estudo realizado por Cynthia Selby (Selby, 2014), a autora relaciona a taxonomia de Bloom com o pensamento computacional e o ensino de programação a partir de entrevistas, questionários, discussões em uma comunidade *online*, com 255 participantes, sendo a maioria professores de áreas associadas à computação ou à ciência da computação. Selby estabelece uma sequência de ensino utilizada pela maioria dos professores como sendo:

1. Construções, fatos, tipos;
2. Funcionamento de construções individuais;
3. O uso de programação construindo contextos artificiais;
4. Discriminar, decompor, abstrair;
5. Criar programas, projetar algoritmos;
6. Testar, avaliar.

As competências eleitas por Selby para caracterizar o pensamento computacional são aqui listadas em ordem crescente de dificuldade de aprendizagem pelos alunos, segundo os pesquisados. Selby utiliza a taxonomia de Bloom de 1956 e relaciona-a com as competências do pensamento computacional e o Ensino de Programação conforme mostra a
Tabela 24.

Tabela 24 - Mapeamento da taxonomia de Bloom, Pensamento Computacional e Ensino de Programação

Taxonomia Bloom 1956	Competências do Pensamento Computacional	Ensino de Programação
Conhecimento		Estruturas, construções, fatos, tipos;
Compreensão		
Aplicação		Discriminar, decompor, abstrair
Análise	Abstração de funções Abstração de dados Decomposição	
Síntese	Projeto de algoritmo	Criar programas, projetar algoritmos
Avaliação	Avaliação	Testar, avaliar

Pelo estudo de Selby deve-se chegar aos três níveis mais altos da taxonomia de Bloom para se atingir o pensamento computacional.

3.4.4 Métricas de *Software*

Para avaliar as produções dos alunos de forma mais objetiva, recorri às métricas de software, técnica amplamente utilizada na área de engenharia de software. Sendo assim,

apresento as métricas de Linhas de Código (*Line of Code*) e Complexidade Ciclomática de McCabe.

3.4.4.1 Linhas de código

Linhas de Código se referem ao código fonte, ao programa escrito pelo usuário. Esta métrica está classificada como uma métrica de comprimento, ou seja, de extensão do trabalho. Ela costuma ser abreviada para LOC a partir do nome em inglês – Line Of Code (LOC), sendo também utilizada a abreviatura KLOC, se referindo a um grupo de mil (K) linhas de código. É uma métrica básica e a partir dela se definem outras como métrica de produtividade (linhas de código geradas por semana) ou de qualidade (defeitos por linha de código).

Quando os programas eram feitos em linguagem de máquina (Assembler) cada linha correspondia a um comando e a uma instrução a ser executada pela máquina. Com o advento das linguagens de alto nível esta relação de um-para-um não existe mais. Além disso, de acordo com cada linguagem e com o estilo de programação do desenvolvedor de *software* esta métrica pode variar. Mesmo a definição do que é uma linha de código no programa fonte pode variar. Por exemplo, na linguagem Pascal, conforme se considerem apenas as linhas executáveis ou as linhas executáveis e as definições de variáveis, contaremos os comentários, ou simplesmente os demarcadores de comandos, como o sinal de ponto-e-vírgula (;) (Kan, 2002). Stephen Kan recomenda que se defina qual a interpretação de linha de código que se está utilizando e se utilize esta métrica criando o seu próprio histórico, associado a um contexto específico. Neste trabalho, estou usando como métrica a contagem dos comandos do Scratch que são arrastados e blocados para formar o programa.

3.4.4.2 Complexidade ciclomática de McCabe

Esta métrica foi proposta por Thomas McCabe em 1976 (McCabe, 1976). É considerada uma métrica da complexidade de *software*. Ela relaciona o fluxo do programa com um gráfico equivalente. Vamos tomar como exemplo um programa com duas estruturas SE-SENÃO, conforme Figura 6.

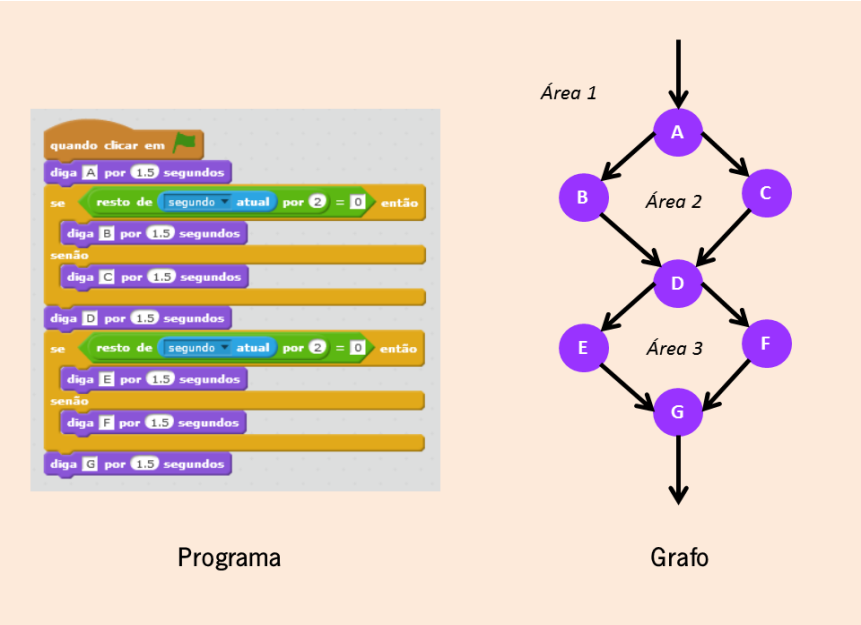


Figura 6 - Exemplo de Programa e Gráfico Equivalente

Esta métrica pode ser definida como a complexidade ciclomática do gráfico equivalente. Em termos de programação equivale aos caminhos linearmente independentes do fluxo do programa. Pode ser quantificado contando-se as áreas independentes isoladas pelo gráfico ou contando o número de desvios do programa mais um. No nosso exemplo, são três as áreas isoladas e são dois os desvios realizados pelos dois comandos SE-SENÃO (Kan, 2002).

Esta métrica tem sido utilizada para identificar os caminhos linearmente independentes para efeito de teste de software. Apesar da complexidade de McCabe apresentar uma correlação com as linhas de código, Kan (2002) nota que a relação de defeitos encontrados em um software apresenta uma relação mais forte com a métrica de McCabe do que com as Linhas de Código. No caso deste trabalho, a métrica de McCabe será definida como a contagem do número de ocorrências dos comandos e desvios relacionados conforme a Figura 7 documenta.

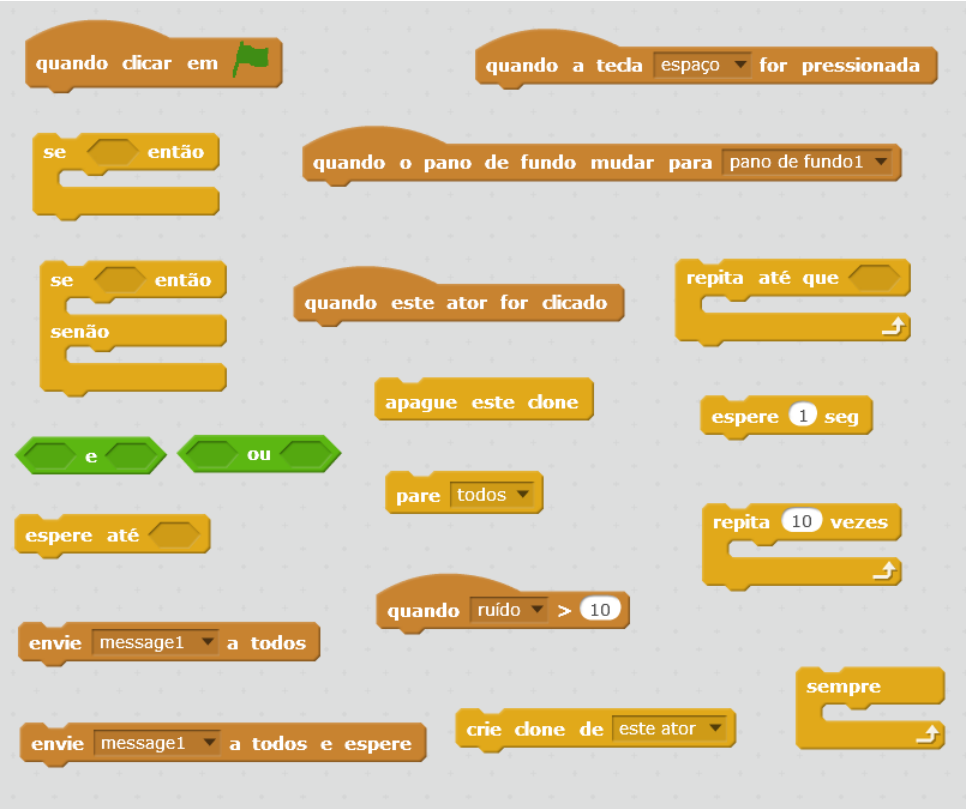


Figura 7 - Comandos de Desvio para Complexidade Ciclomática de McCabe

4 Opção metodológica

Iniciei a lecionação da disciplina de Introdução à Lógica de Programação em 2010 na Universidade Federal do Recôncavo da Bahia (UFRB). Nessa altura usava-se a linguagem C ou Pascal que possuíam interface texto. O curso estava baseado em ensinamentos de algoritmos que eram memorizados e reproduzidos nas provas com pouca ou nenhuma modificação. Neste contexto os alunos se mostravam desmotivados e muito estressados nos laboratórios, com os erros de sintaxe, sem mostrar autonomia ou criatividade no final do curso. Eu gostaria que os alunos trabalhassem em uma interface gráfica, que pudessem produzir seus programas mostrando criatividade e autonomia, utilizando uma programação orientada a objetos. Gostaria de explorar a matemática básica nas aulas de programação, gostaria que esta disciplina contribuísse de maneira mais efetiva para a formação desses alunos, sendo útil e interessante. A partir dessa situação problemática, formulei uma pergunta de pesquisa:

Como conduzir um processo de aprendizagem em salas regulares de programação de computadores, de modo a contribuir para o desenvolvimento do pensamento computacional na formação de futuros professores de matemática?

Minha intenção era contribuir para mudar a situação descrita, tornando o curso de Introdução à Lógica de Programação mais interessante e útil. Para isso, experimentei atividades em sala de aula que deram a oportunidade aos alunos de serem criativos e autônomos.

Minhas contribuições para o ensino de programação estão relacionadas com as atividades e os processos desenvolvidos em sala de aula, podendo gerar mudança de comportamento nos alunos, tornando-os mais engajados no aprendizado, ultrapassando a simples memorização de técnicas de programação.

O foco da investigação são as contribuições para a formação destes alunos que serão futuros professores de matemática, embora as práticas possam ser úteis a outros professores e a outros alunos. No meu estudo foquei as minhas leituras no aprendizado em geral e no aprendizado de matemática em particular. Entre os alunos que participaram da pesquisa, incluí os alunos de licenciatura em física, além dos licenciandos em matemática, uma vez que utilizei as mesmas práticas e espero os mesmos resultados com relação à aprendizagem.

A pesquisa tem como objetivo ajudar a mudar as práticas de ensino, tornando as aulas mais úteis e interessantes para os alunos. Pesquisei outras possibilidades de ensino da

programação, com outras ferramentas que facilitam e possibilitam a criatividade e a autonomia do aluno com relação à programação usando as concepções de ensino e os conteúdos da área de ensino de matemática, assim como utilizando a programação orientada a objetos como recomendada pela área de computação. Considerarei contribuições as práticas que se mostrarem vantajosas. Querendo que a investigação aponte as práticas promissoras utilizei a seguinte estratégia:

- Pesquisa de referências acadêmicas e possibilidades de mudança.
- Mudança das práticas adaptando as sugestões da literatura ao contexto do curso.
- Avaliação do impacto das mudanças.

4.1 Contexto da investigação

Participaram neste trabalho os alunos que cursaram Licenciatura em Matemática e Licenciatura em Física da UFRB entre os anos de 2011 a 2014, porque é com estes alunos que experimentei as práticas de que dou conta agora.

O curso de licenciatura em matemática da UFRB ocorre no Centro de Formação de Professores no município de Amargosa, no estado da Bahia. Para contextualizar o ambiente da investigação utilizei os índices Índice de Desenvolvimento Humano (IDH) e o Programa Internacional de Avaliação de Estudantes (PISA). Estes índices não são capazes de abarcar toda a realidade e complexidade da educação e dos dados socioeconômicos, de uma população, mas servem como uma janela, ainda que estreita, para se ter uma visão comparativa da realidade do ambiente investigado.

O IDH surgiu como um contraponto ao índice de renda *per capita* que avalia os países, considerando apenas o aspecto econômico, sendo um indicador mais adequado para refletir o desenvolvimento humano das populações (O. M. P. Silva & Panhoca, 2007). O IDH fornece uma síntese de três indicadores coletados com facilidade, que abrangem três aspectos universais da vida humana: a possibilidade de escolhas e oportunidades refletindo-se na educação; o direito a uma vida longa e saudável refletida na longevidade; no direito a um padrão de vida digno, refletido na renda (Bitoun, 2005). Escolhi este índice para que o leitor distante da realidade do interior da Bahia tenha um parâmetro de comparação relacionando as condições de vida de países ou populações mais próximas com índices parecidos.

O PISA é desenvolvido pela Organização para Cooperação e Desenvolvimento Econômico (OCDE). Este programa promove uma avaliação padronizada de estudantes em 64 países com aproximadamente 15 anos de idade. São avaliadas as áreas de leitura, ciências e matemática (OCDE, 2012).

O PISA enfatiza a avaliação das competências escolares necessárias à vida moderna, sem se deter somente aos conteúdos curriculares. A avaliação do PISA procura verificar a operacionalização de esquemas cognitivos através da capacidade do aluno aplicar seus conhecimentos, analisar, raciocinar e se comunicar com eficiência, à medida que expõe, resolve e interpreta problemas, em situações diversas. É uma tentativa de medir a efetividade dos sistemas educativos de diversos países e possibilitar a comparação entre eles (Silva Aguiar & Ortigão, 2012).

A cada edição uma das áreas é enfatizada. Na edição de 2012, a área de matemática foi enfatizada com 54% dos itens de avaliação voltados a matemática, 23% dos itens voltados à leitura e 23% para as ciências. Para este trabalho utilizarei os índices do PISA específicos da área de matemática (OCDE, 2012).

4.1.1 O Estado da Bahia

A Bahia é um estado com o IDH de 0,660 e PISA de 373,2. Comparando a Bahia com as demais 27 unidades federativas do Brasil, este índice coloca a Bahia em 23.º lugar, se ordenarmos as unidades federativas do Brasil pelo IDH do melhor ao pior índice e em 17.º lugar se ordenamos as unidades federativas pelo PISA do melhor ao pior índice.

O Distrito Federal é aquele que apresenta os melhores índices de IDH e PISA, 0,824 e 415,8 respectivamente, e Alagoas os piores IDH e PISA, 0,639 e 324,2 respectivamente. Ver aTabela 25.

Tabela 25 - Valores de IDH e PISA por Unidade Federativa do Brasil

Item	UF	IDH	PISA
1	Distrito Federal	0,824	415,8
2	Santa Catarina	0,774	415,3
3	Espírito Santo	0,74	414,2
4	Mato Grosso do Sul	0,729	408,3
5	Rio Grande do Sul	0,746	407,0
6	São Paulo	0,783	403,6
7	Paraná	0,749	403,5
8	Minas Gerais	0,731	403,1
9	Paraíba	0,658	395,3
10	Rio de Janeiro	0,761	388,8
11	Piauí	0,646	385,3
12	Sergipe	0,665	384,0
13	Rondônia	0,690	381,9
14	Rio Grande do Norte	0,684	380,4
15	Goiás	0,735	379,1
16	Ceará	0,682	378,3
17	Bahia	0,660	373,2
18	Mato Grosso	0,725	370,2
19	Tocantins	0,699	365,5
20	Pernambuco	0,673	363,4
21	Roraima	0,707	361,8
22	Amapá	0,708	360,2
23	Pará	0,646	359,8
24	Acre	0,663	358,7
25	Amazonas	0,674	355,8
26	Maranhão	0,639	343,2
27	Alagoas	0,631	342,0

Fonte:(OCDE, 2012; D. G. Pinto, Costa, & Marques, 2013).

Observa-se certa correlação entre o IDH e o PISA, considerando as unidades federativas no Brasil, como mostra a Figura 8.

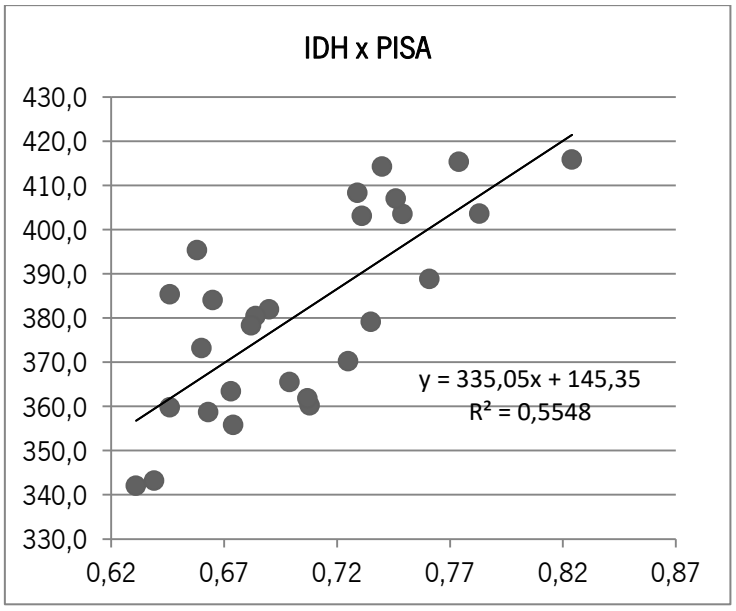


Figura 8 - IDH versus PISA das Unidades Federativas do Brasil

4.1.2 O perfil do aluno da UFRB

Frequentaram Introdução à Lógica de Programação 226 alunos, realizando todas as tarefas até o projeto final. Os alunos se distribuíam por 5 turmas de licenciatura em Física e 5 turmas de licenciatura em matemática. As aulas eram ministradas em 2 encontros semanais de 2 horas cada, durante 15 semanas, totalizando 60 horas por turma e 600 horas considerando as 10 turmas. Entre os alunos, 55% eram mulheres (ver Figura 9) e 88% frequentaram escola pública (ver Figura 10). As idades variaram de 16 a 50 anos, mas 91% dos alunos estavam abaixo de 30 anos, a média de idade é de 22 anos e a idade mais frequente é a de 19 anos (ver Figura 11).

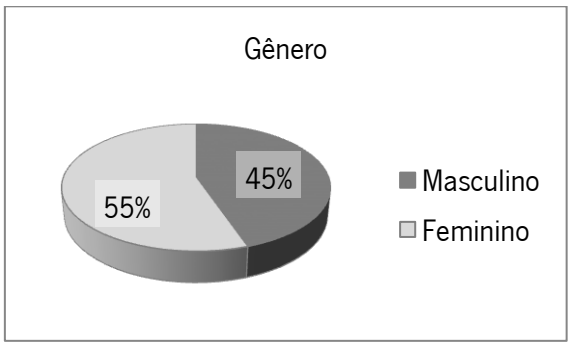


Figura 9 - Gráfico Gênero dos Alunos da Amostra

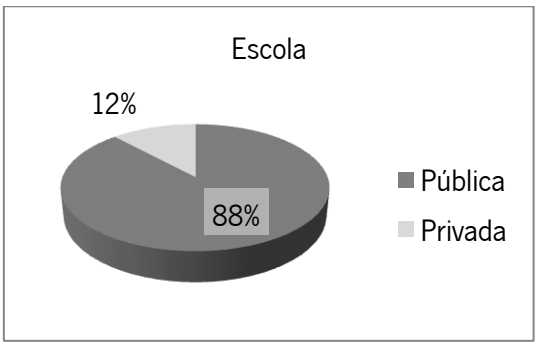


Figura 10 - Gráfico Tipo da Escola de Origem

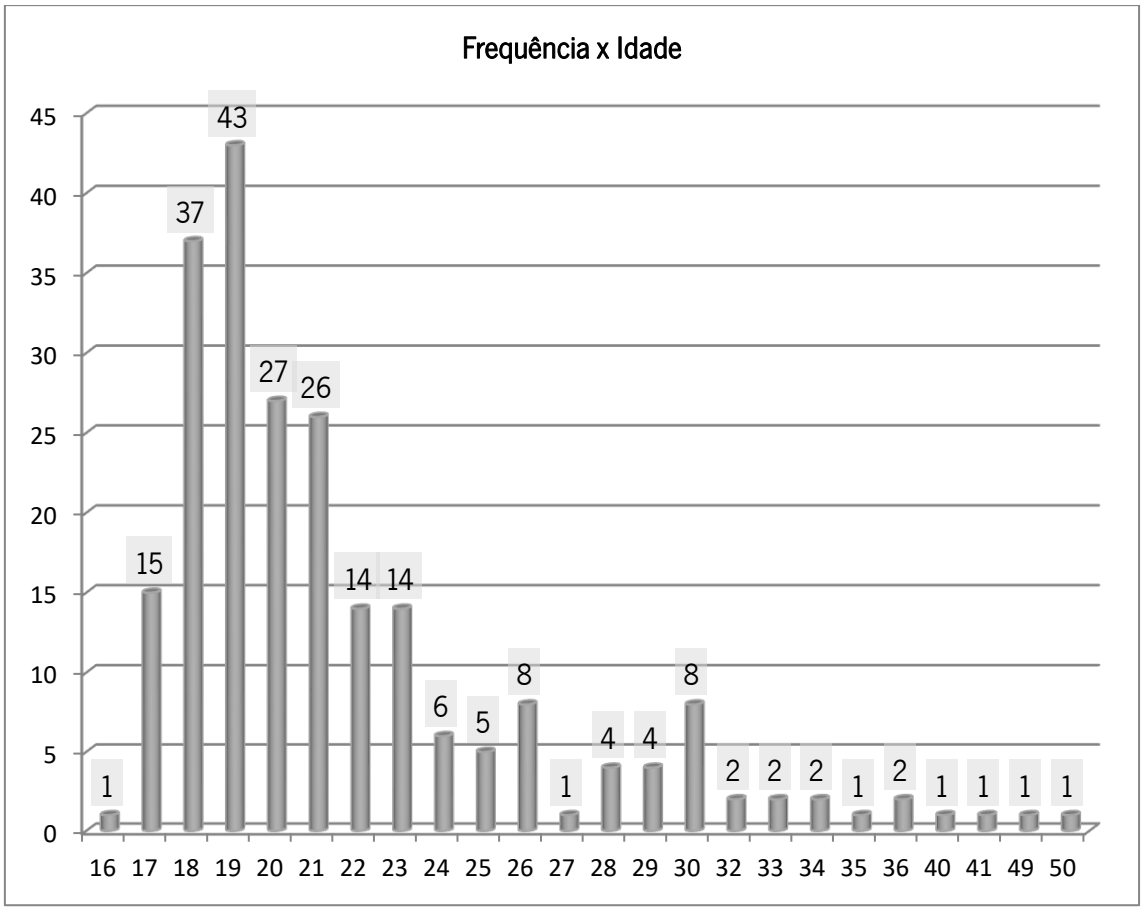


Figura 11 - Gráfico de Frequência das Idades dos Alunos

Quanto ao município de origem, isto é, o município no qual foi concluído o Ensino Médio, o Índice de Desenvolvimento Humano Municipal (IDHM) se encontra entre 0,805 e 0,549 sendo a média considerando todos estes municípios de 0,640 (ver Figura 12). Utilizando as faixas de classificação brasileira, 66% dos municípios da amostra estão classificados com IDH médio, 18% apresentam IDHM baixo e 15% alto (ver Figura 13).

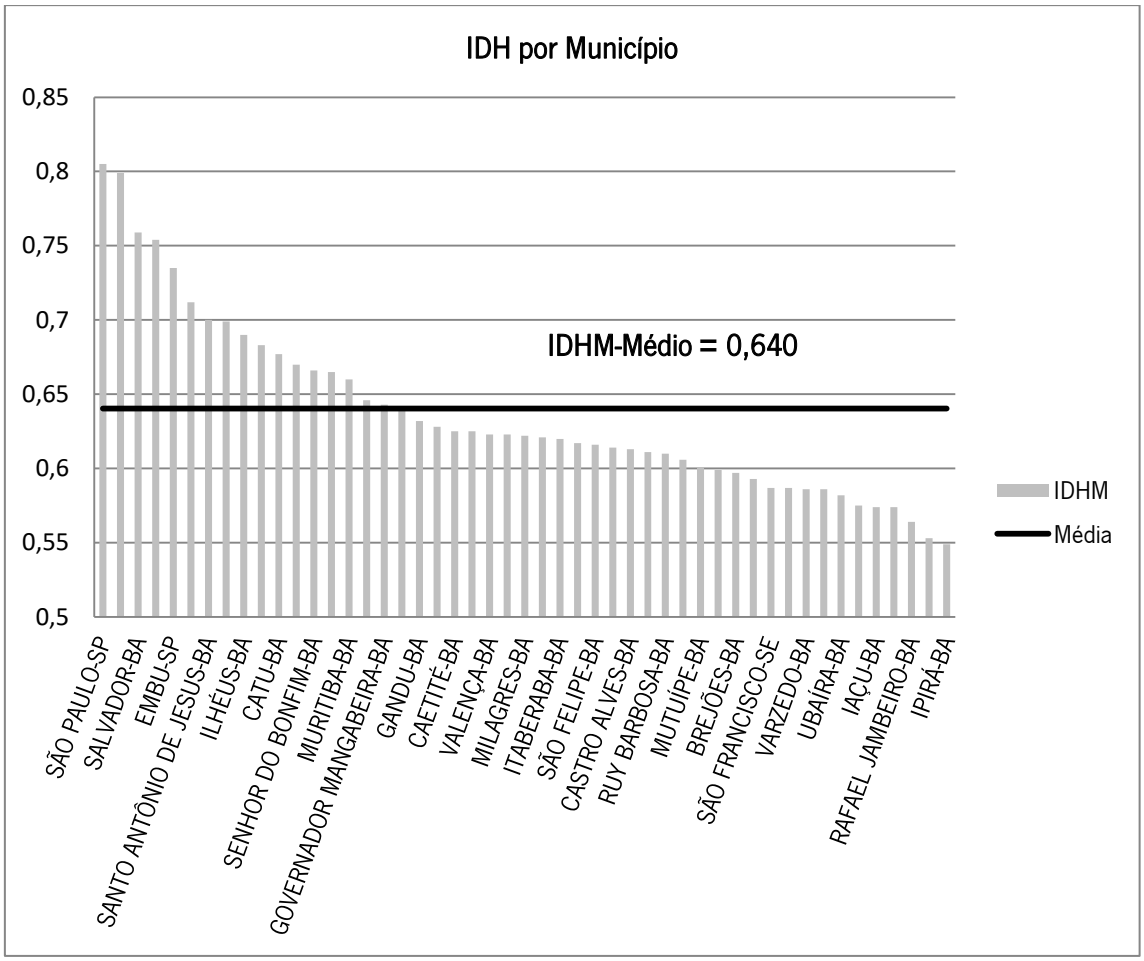


Figura 12 - IDH por Município

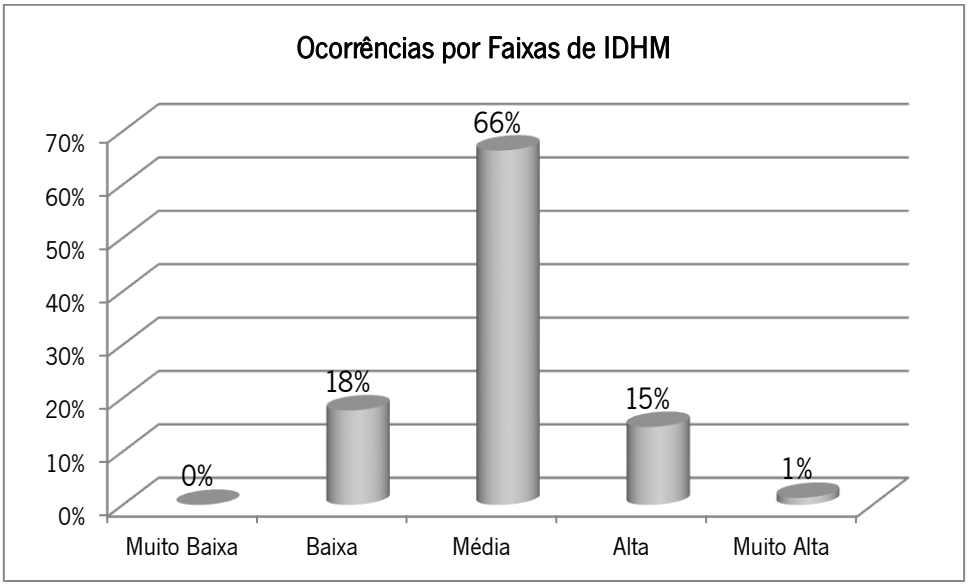


Figura 13 - Ocorrências por Faixa de IDHM

Não existe um índice PISA por município, mas se considerarmos a tendência nacional da correlação entre PISA x IDH, pode-se estimar que o PISA médio em função do IDHM-Médio deveria ser próximo de 359,8 (PISA Municipal Médio = 359,8). Usando a mesma correlação para cada município da nossa amostra, pode-se inferir que 67% dos municípios se encontram abaixo do nível 1 da classificação do PISA e 33% se encontram no nível 1 do PISA (ver Figura 14).

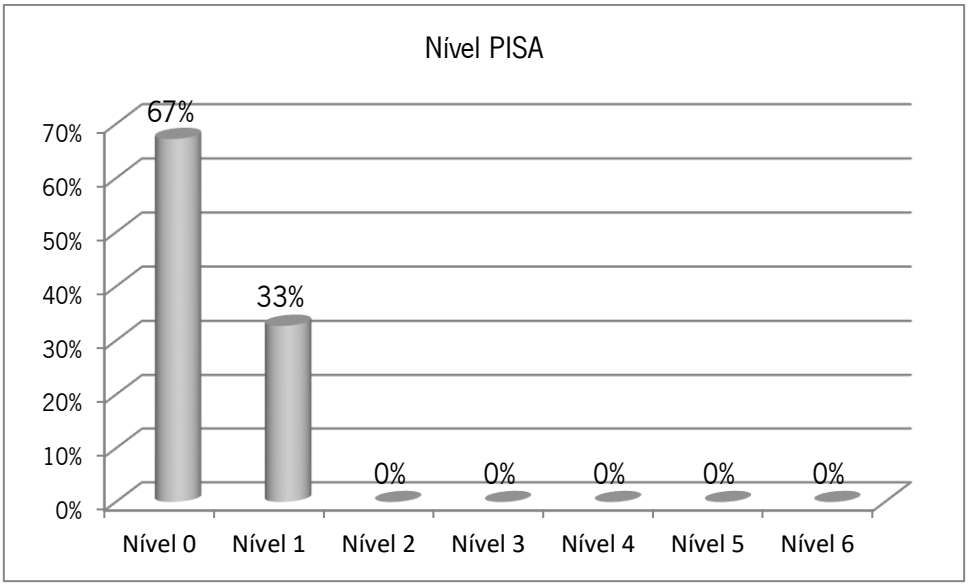


Figura 14 - Ocorrências dos Níveis do PISA na Amostra

A OCDE define que o nível 1 tem a seguinte característica:

> No Nível 1, os estudantes são capazes de responder a questões definidas com clareza, que envolvem contextos conhecidos, nas quais todas as informações relevantes estão presentes. Conseguem identificar informações e executar procedimentos rotineiros de acordo com instruções diretas em situações explícitas. São capazes de executar ações óbvias e dar continuidade imediata ao estímulo dado. (OCDE, 2012, p. 19)

Eu gostaria de acrescentar alguns comentários coerentes com os baixos índices de PISA e IDH. Nossos alunos vêm, predominantemente, de escolas públicas que estão desprivilegiadas pelas políticas públicas e carregam o estigma de um ensino ruim, reforçado pelos baixos índices de efetividade. No primeiro semestre de matemática existe a disciplina de Introdução ao Cálculo, que seria teoricamente uma revisão da matemática do ensino médio. No entanto, os alunos relatam dificuldades na disciplina e afirmam que, em muitas das aulas, é o seu primeiro contato com o conteúdo ensinado. Em termos gerais concluo que o aluno típico é o de baixa renda, que

estudou em escola pública e tem um défice educacional, quando comparado com as competências e habilidades esperadas no final do ensino médio.

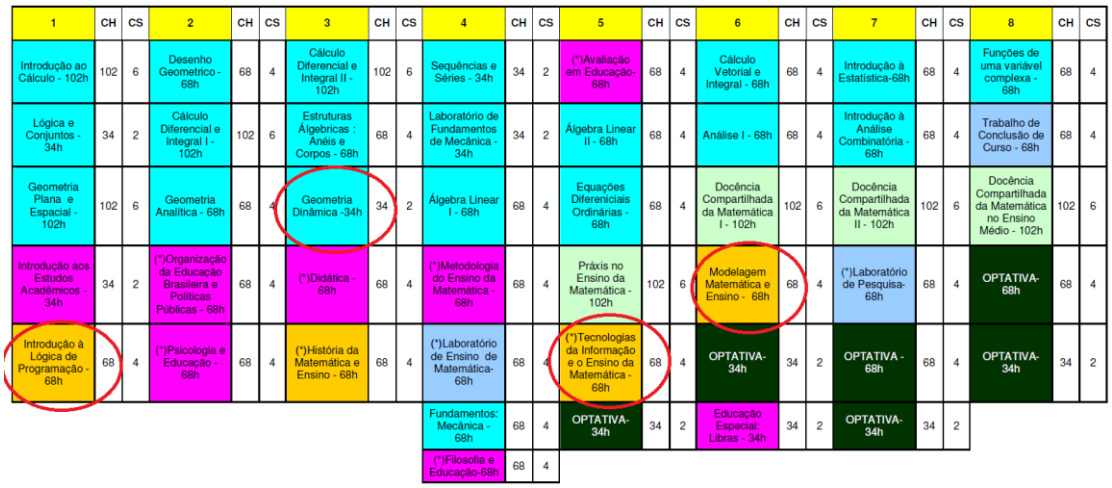
4.1.3 O curso de licenciatura em Matemática

O curso de licenciatura em matemática tem o objetivo de formar professores conscientes da cidadania, críticos e criativos, preparado para as mudanças de contexto do futuro. O Projeto Pedagógico do Curso de Licenciatura em Matemática da UFRB (Coordenadoria de Ensino e Integração Acadêmica, 2007) define o seguinte perfil do seu egresso:

> O curso deve garantir que seus egressos tenham uma sólida formação de conteúdos matemáticos e pedagógicos que possibilite tanto à vivência crítica da realidade do ensino fundamental e médio como também a experimentação de novas propostas que considere a evolução dos estudos da Educação Matemática, bem como uma formação geral complementar envolvendo outros campos do conhecimento necessários ao exercício do magistério.
> Por isso, o egresso do Curso de Licenciatura em Matemática deverá ter uma formação que o prepare para enfrentar os desafios das rápidas transformações da sociedade e das condições de exercício profissional, com domínio dos conteúdos a serem socializados e consciência do seu papel de educador tendo autonomia, responsabilidade, comprometimento com a sua atuação, condições de avaliar e utilizar novas tecnologias de ensino, assim Contribuindo para o exercício da cidadania. (Coordenadoria de Ensino e Integração Acadêmica, 2007, p. 13)

O curso tem uma carga horária total de 3192 horas, sendo dividida em 2312 horas, em disciplinas obrigatórias, 272 horas em disciplinas optativas, 408 horas de estágio e 200 horas em atividades complementares. Das 2312 horas em disciplinas obrigatórias, 238 (10,3%) são utilizadas em disciplinas que usam sistematicamente computadores e estimulam o pensamento computacional. São elas, “Introdução a lógica de Programação”, “Geometria Dinâmica”, “Tecnologias de Informação e o Ensino da Matemática” e também “Modelagem Matemática e Ensino” (ver Figura 15). Em minha avaliação, as críticas que se fazem aos cursos de licenciatura em matemática por não ensinarem tecnologias voltadas ao ensino não procedem no caso da UFRB. Ainda assim, há espaço para aumentar esta opção pelas tecnologias, mas acredito que deva ser feita de maneira interdisciplinar, na medida das necessidades das outras disciplinas.

Matriz Curricular



1	CH	CS	2	CH	CS	3	CH	CS	4	CH	CS	5	CH	CS	6	CH	CS	7	CH	CS	8	CH	CS
Introdução ao Cálculo - 102h	102	6	Desenho Geometrico - 68h	68	4	Cálculo Diferencial e Integral II - 102h	102	6	Sequências e Séries - 34h	34	2	(*)Avaliação em Educação- 68h	68	4	Cálculo Vetorial e Integral - 68h	68	4	Introdução à Estatística-68h	68	4	Funções de uma variável complexa - 68h	68	4
Lógica e Conjuntos - 34h	34	2	Cálculo Diferencial e Integral I - 102h	102	6	Estruturas Álgebricas : Anéis e Corpos - 68h	68	4	Laboratório de Fundamentos de Mecânica - 34h	34	2	Álgebra Linear II - 68h	68	4	Análise I - 68h	68	4	Introdução à Análise Combinatória - 68h	68	4	Trabalho de Conclusão de Curso - 68h	68	4
Geometria Plana e Espacial - 102h	102	6	Geometria Analítica - 68h	68	4	Geometria Dinâmica -34h	34	2	Álgebra Linear I - 68h	68	4	Equações Diferenciais Ordinárias - 68h	68	4	Docência Compartilhada da Matemática I - 102h	102	6	Docência Compartilhada da Matemática II - 102h	102	6	Docência Compartilhada da Matemática no Ensino Médio - 102h	102	6
Introdução aos Estudos Acadêmicos - 34h	34	2	(*)Organização da Educação Brasileira e Políticas Públicas - 68h	68	4	(*)Didática - 68h	68	4	(*)Metodologia do Ensino da Matemática - 68h	68	4	Práxis no Ensino da Matemática - 102h	102	6	Modelagem Matemática e Ensino - 68h	68	4	(*)Laboratório de Pesquisa- 68h	68	4	OPTATIVA- 68h	68	4
Introdução à Lógica de Programação - 68h	68	4	(*)Psicologia e Educação - 68h	68	4	(*)História da Matemática e Ensino - 68h	68	4	(*)Laboratório de Ensino de Matemática- 68h	68	4	(*)Tecnologias da Informação e o Ensino da Matemática - 68h	68	4	OPTATIVA- 34h	34	2	OPTATIVA - 68h	68	4	OPTATIVA- 34h	34	2
									Fundamentos: Mecânica - 68h	68	4	OPTATIVA- 34h	34	2	Educação Especial: Libras - 34h	34	2	OPTATIVA- 34h	34	2			
									(*)Filosofia e Educação-68h	68	4												

Figura 15 - Imagem da Grade Curricular (Coordenadoria de Ensino e Integração Acadêmica, 2007)

4.2 Projeto de intervenção

Sendo a questão de investigação tão subjetiva e sendo tão próximo o investigador e o objeto de pesquisa, me distancio do paradigma positivista. O paradigma interpretativo está mais próximo das minhas intenções, servindo para a observação participativa das minhas aulas e para a interpretação da cultura vigente entre os alunos. Clara Coutinho refere a busca de significado profundos do comportamento na interação humana, com suporte na hermenêutica (Coutinho, 2013). O investigador e investigado interagem sob regras socioculturais buscando se interpretar mutuamente em um processo de dupla hermenêutica. Os desafios culturais se tornam complexos ao considerar a cultura local em uma geração globalmente conhecida como Nativos Digitais (Prensky, 2000), cujo modo de se relacionar com o mundo é diferente do de seus professores, de uma era pré-digital e da instituição escolar com algumas práticas do século XIX.

Este trabalho não se limita à compreensão, mas também possui um caráter intervencionista. Pretendo incentivar as competências indicadas por Rush e outros como necessárias ao século XXI (Maloney et al., 2010) e assim, causar uma mudança de comportamento no aluno e na postura do professor para que haja um ambiente motivador da aprendizagem incentivando a criatividade. Deste modo, o paradigma sociocrítico se mostra o mais adequado à investigação. Segundo Coutinho (2013) o paradigma sociocrítico é baseado na filosofia marxista e tem Paulo Freire como expoente na área da educação, com a sua pedagogia

da libertação, tendo a intenção de modificar o mundo rumo à liberdade, justiça e democracia. A teoria crítica assume que não há uma perspectiva neutra na investigação. O conhecimento é uma construção social que tem um interesse técnico, comunicacional ou crítico emancipatório. O paradigma sociocrítico possui semelhanças metodológicas com o paradigma qualitativo, mas inclui o componente ideológico de intervenção da realidade presente em estudos de currículo, administração educacional e formação de professores (Coutinho, 2013).

O conceito de paradigma foi criado em 1962 por Thomas Kuhn e definido como sendo "o conjunto de crenças, valores e técnicas partilhadas pelos membros de uma dada comunidade científica e como um modelo para o 'que' e para o 'como' investigar num dado e definido contexto histórico/social". Já a metodologia cuida dos princípios, limites e adequação dos métodos à investigação. Em termos metodológicos a teoria crítica se serve de métodos tanto quantitativos quanto qualitativos. (Coutinho, 2013).

A questão de pesquisa remete para a descoberta das contribuições da programação de computadores a partir da prática, caracterizando o modo indutivo de produzir conhecimento, tendo como foco a práxis do professor, isto é, a ação ordenada sobre o processo de ensino-aprendizagem, interferindo e sendo influenciado pelo contexto social de forma dialética.

Nesta investigação, me propus a realizá-la em duas etapas. A primeira para definir, de modo geral, uma práxis de ensino de programação com Scratch, usando a autobiografia como técnica de recolha de dados, analisando os dados com base nos parâmetros estabelecidos pela ferramenta Dr. Scratch de avaliação do pensamento computacional. Como produto dessa primeira etapa, terei uma proposta de estrutura para o curso e algumas contribuições. A estrutura do curso e suas contribuições serão validadas em uma segunda etapa, que serviu de triangulação, na qual a análise documental, visando evidenciar o processo de aprendizagem adotado, como técnica de recolha de dados. A análise e validação da práxis e suas contribuições são analisadas segundo a taxonomia de Bloom.

4.2.1 Metodologia de Estudo de Caso

Consideraremos a disciplina Introdução a Lógica de Programação como o caso a ser estudado. Compilando opiniões de vários autores, Coutinho destaca cinco atitudes que o investigador de estudos de casos deve assumir:

- A primeira tarefa do investigador é pois definir as fronteiras do caso da forma mais clara precisa.

- Segundo, é um caso sobre "algo", que há que identificar para conferir foco e direção à investigação.
- Terceiro, tem de haver sempre a preocupação de preservar o caráter 'único, específico, diferente, complexo do caso'; a palavra holístico é muitas vezes usada nesse sentido.
- Quarto, a investigação decorre em ambiente natural.
- Quinto, o investigador recorre a fontes múltiplas de dados e a métodos de recolha muito diversificados. (Coutinho, 2013, p. 335)

Vamos tomar este roteiro como modelo para justificar a escolha do estudo de casos.

Fronteiras – O estudo de caso se limitará a turmas de Introdução a Lógica de Programação dos cursos de Licenciatura em Física e Licenciatura em Matemática, em 4 anos, iniciando no ano de 2011 até o ano letivo de 2014.

Foco – O objetivo é descrever e avaliar as contribuições de uma práxis que tema intenção de contribuir para formação de futuros professores, nos aspectos sociais e cognitivos referente aos conteúdos de matemática, programação e pedagógicos, bem como aspectos relativos à criatividade e resolução de problemas.

Especificidade – Este caso, apesar de se referir às classes identificadas anteriormente (fronteiras), trata do ensino de programação adaptado para essa nova geração de Nativos Digitais, explorando as relações entre a cultura dessa nova geração e os conteúdos acadêmicos, incentivando a mudança de atitude para uma conduta emancipatória.

Ambiente natural – Os casos estudados são de turmas regulares da UFRB.

Fontes Múltiplas de Dados – Os dados foram coletados nos anos de 2011 a 2014 para uma análise documental dos trabalhos realizados neste período.

Pode-se classificar o estudo de caso da investigação como sendo um caso único e global. Generalizando as diversas turmas e assim considerando a disciplina de Introdução a Lógica de Programação como o caso a ser estudado, pode-se considerar como um estudo de caso único. Os dados obtidos de diferentes anos e turmas serão considerados como uma única unidade de análise, tomando-os como estudo de caso global (Coutinho, 2013).

Coutinho destaca as possibilidades do estudo de caso na pesquisa científica como sendo para explorar, descrever, explicar, avaliar e/ou transformar.

4.2.1.1 A técnica autobiográfica

A técnica que utilizei para recolha dos dados é a autobiografia. A autobiografia está relacionada ao método autobiográfico e atende as metodologias qualitativas. Esta metodologia é

um dos métodos da abordagem biográfica, junto com biografia, história de vida e história oral. A abordagem biográfica surgiu com o movimento conhecido como Escola de Chicago, referindo-se aos trabalhos de psicologia e sociologia desenvolvidos entre 1913 e 1940, na Universidade de Chicago, fundada um pouco antes em 1890 (O. M. P. Silva & Panhoca, 2007). Suas características principais são:

> (1) o discurso direcionado ao leitor, levando muitas vezes uma não-reflexão sobre o vivido;
> (2) a preocupação com a rememoração das experiências pessoais e sua articulação num contexto histórico mais amplo destacando assim uma preocupação com a sequência temporal;
> (3) o uso da descrição para revelar os momentos de sua história. (O. M. P. Silva & Panhoca, 2007, p. 29)

No Brasil, os métodos biográficos e autobiográficos aparecem a partir de 1990, com um aumento significativo a partir de 1995 (Bueno, Chamlian, Sousa, & Catani, 2006). As publicações portuguesas de António Nóvoa *O Método (Auto)biográfico e a Formação* de 1988 e duas coletâneas organizadas por ele em 1995 intituladas *Vida de Professor* e *Profissão Professor*, influenciaram as pesquisas brasileiras. Bueno também assinala o caráter introspectivo da autobiografia afirmando que enquanto relato oral é uma palavra endereçada e atenta às reações do receptor (Bueno et al., 2006).

Neste trabalho utilizarei o relato autobiográfico ao serviço do projeto de pesquisa. A partir dos dados e à luz do referencial teórico, devo chegar a uma práxis que influenciará os envolvidos (professor e alunos), caracterizando o modo indutivo de geração de conhecimento.

4.2.1.2 Avaliação dos produtos do curso

No final dessa primeira etapa a autobiografia deve relatar como se deu a construção da estrutura do curso e algumas contribuições observadas. A estrutura geral do curso ficou dividida em três tipos de atividades a saber: Roteiros, Provas e Projetos.

A avaliação inclui a submissão de uma amostra de programas ao site Dr. Scratch, uma vez que este site oferece uma ferramenta que avalia os programas, identificando as estruturas de programação que caracterizam aspectos como abstração e decomposição de problemas, paralelismo, pensamento lógico, controle de fluxo, interatividade com o usuário, representação dos dados. Estes sete aspectos são pontuados de 0 a 3, dependendo da existência de certas estruturas de programação. Juntando cada um dos valores dos sete aspectos, obtém-se um valor de proficiência geral do programa desenvolvido em Scratch, que pode variar de 0 a 21.

Foram escolhidos para a amostra 10 programas, cujo critério de escolha será mencionado mais adiante neste capítulo.

4.2.1.3 Análise documental

A pesquisa documental pode se referir tanto ao método como à técnica de pesquisa (Sá-Silva, Almeida, & Guindani, 2009). Neste trabalho usarei a pesquisa documental como técnica de análise, com o objetivo de avaliar o processo de aprendizagem. Nesse sentido, a pesquisa documental é recomendada por Coutinho (2013) como uma técnica de recolha de dados para análise documental. A análise é feita a partir de documentos escritos, como cartas, memorandos, comunicados, agendas, planos, propostas, cronogramas, jornais internos e outros. Neste trabalho, as tarefas propostas (roteiros, provas e projetos) se enquadram nessa definição. Mas, para considerar os programas como documentos, que evidenciam os trabalhos realizados pelos alunos, precisa-se de uma concepção mais ampla do que é um documento. O movimento da Escola de Annales em 1950 amplia o conceito de documento considerando "tudo o que, pertencendo ao homem, depende do homem, serve o homem, exprime o homem, demonstra a presença, a atividade, os gostos e as maneiras de ser do homem", referem Sá-Silva et al. (2009). Nesta nova concepção, passam a ser considerados documentos, não apenas os documentos escritos, mas também os achados arqueológicos, as imagens, as rochas utilizadas pelos geólogos, os metais utilizados pela química e outros tantos objetos em suportes variados.

Os programas de computadores podem ser considerados como documentos digitais, como define Fábio de Almeida.

> Documento digital é aquele documento – de conteúdo tão variável quanto os registros da atividade humana possam permitir – codificado em sistema de dígitos binários, implicando na necessidade de uma máquina para intermediar o acesso às informações. Tal máquina é, na maioria das vezes, um computador. (Almeida, 2011, p. 17)

Almeida acrescenta ainda a seguinte característica dos documentos digitais:

> Os documentos digitais têm como característica a dissociação entre o suporte físico e o seu conteúdo informacional. Sendo assim, é possível o descarte do suporte físico e a manutenção de seu conteúdo em um novo suporte. A evolução extremamente rápida da tecnologia informática torna os suportes físicos de informação obsoletos em um curto espaço de tempo. (Almeida, 2011, p. 16)

As tarefas (roteiros, provas, projetos) e os trabalhos realizados (programas e outros documentos digitais) são considerados dados primários, carecendo de tratamento. A análise, por

sua vez, necessita de uma referência, uma estrutura teórica para que seu conteúdo seja entendido (Sá-Silva et al., 2009). Esta estrutura teórica será definida nos critérios de avaliação.

4.2.1.4 Avaliação do processo de aprendizagem

O objetivo da análise é a avaliação do processo de aprendizagem a partir dos documentos que evidenciam as atividades realizadas. Como referencial teórico para a análise documental, usarei a taxonomia de Bloom. Procurarei, nos documentos, evidências das atividades do processo que possam ajudar a classificar a aprendizagem nas dimensões Conhecimento e Processo Cognitivo da Taxonomia de Bloom. Deste modo, as classes para a análises serão dadas *a priori* pela teoria adotada.

4.2.2 Seleção das amostras de documentos e programas

Para a análise documental, serão observados os documentos gerados nas atividades propostas. As atividades propostas são de três tipos, como já referido: Roteiros, Provas e Projetos. Para realizar uma atividade o professor faz aos alunos uma proposta a que chamaremos "Tarefa". O aluno se debruça sobre a tarefa e realiza um "Trabalho". O "Trabalho" é, portanto, o documento que evidencia a realização da atividade, enquanto a "Tarefa" é o documento que evidencia a proposta feita pelo professor. Veja-se, a este propósito o esquema da Figura 16.

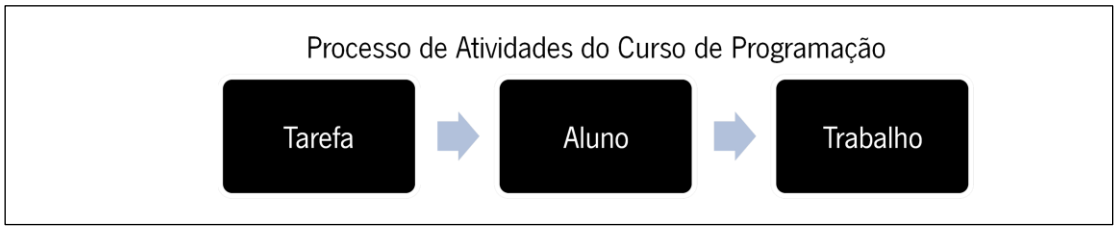


Figura 16 - Modelo Tarefa x Trabalho

Para cada um dos três tipos de atividades, analisarei amostras dos documentos existentes. As provas e os roteiros foram sendo aperfeiçoados ao longo dos anos, de modo que as melhores amostras serão referentes ao ano letivo de 2014.

A realização do projeto final é o ponto alto do curso, visto que, ao realizar o projeto, o aluno atinge o seu auge. Selecionei uma amostra de 10 projetos entre os 88 recolhidos no período de 2011 a 2014. Estes 10 projetos foram selecionados tomando como parâmetro as métricas de extensão e complexidade. A extensão do programa foi medida em linhas de código

(LOC). A complexidade do código foi estabelecida pela complexidade ciclomática de McCabe. Estas duas métricas foram avaliadas e os programas classificados segundo uma distribuição normal. A distribuição normal é um modelo estatístico para eventos em que as ocorrências se acumulam em torno da média. As ocorrências que se afastam da média têm menor probabilidade de acontecer. As métricas de extensão e complexidade obedecem a esta tendência quando adotamos uma escala logarítmica. Isso porque as interações de um programa crescem exponencialmente na medida em que as funções do programa crescem linearmente.

A partir da Média (M) e do Desvio Padrão (σ), podemos tomar alguns pontos de referência que denominaremos de Média Alta (MA), Média Baixa (MB), Média Muito Alta (MMA) e Média Muito Baixa (MMB) (ver Figura 17). A Média Alta e a Média Baixa se distanciam da média em, aproximadamente, 70% do desvio padrão, já a Média Muito Alta e Média Muito Baixa se distanciam da média em, aproximadamente, 160% do desvio padrão (ver Figura 18).

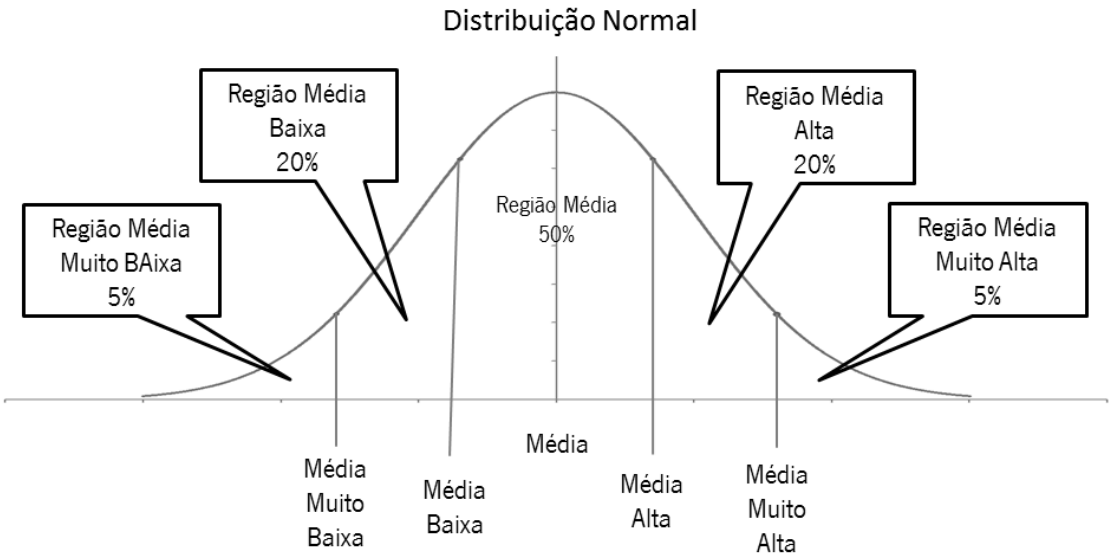


Figura 17 – Modelo de Distribuição Normal Utilizado

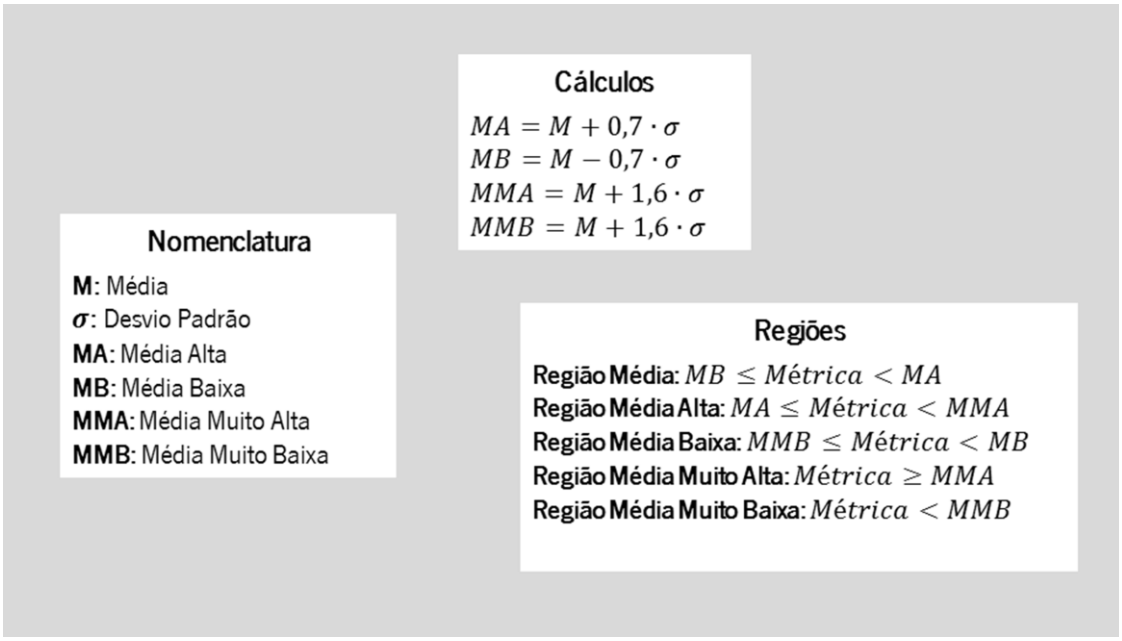


Figura 18 - Definições do Modelo Estatístico Utilizado

A área abaixo da curva normal representa a probabilidade de ocorrência de uma métrica. As métricas entre a Média Alta e a Média Baixa têm 50% de chance de ocorrer, definindo, no gráfico, o que chamei de Região Média. As métricas entre a Média Alta e a Média Muito Alta têm 20% de chance de ocorrer, enquanto as métricas maiores que a Média Muito Alta tem 5% de chance de ocorrer. As métricas entre a Média Baixa e a Média Muito Baixa têm 20% de chance de ocorrer, enquanto as métricas menores que a Média Muito Baixa têm 5% de chance de ocorrer (ver Figura 17).

Usando as definições descritas, separei as classes de frequência e criei os histogramas com os 88 projetos, como mostra a Figura 19 e na Figura 20.

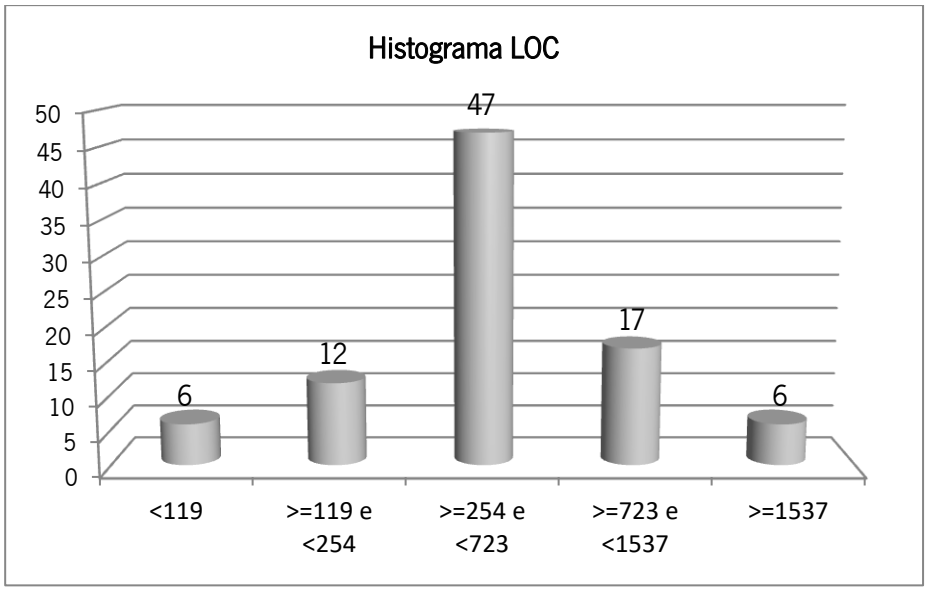


Figura 19 - Histograma de Linhas de Códigos

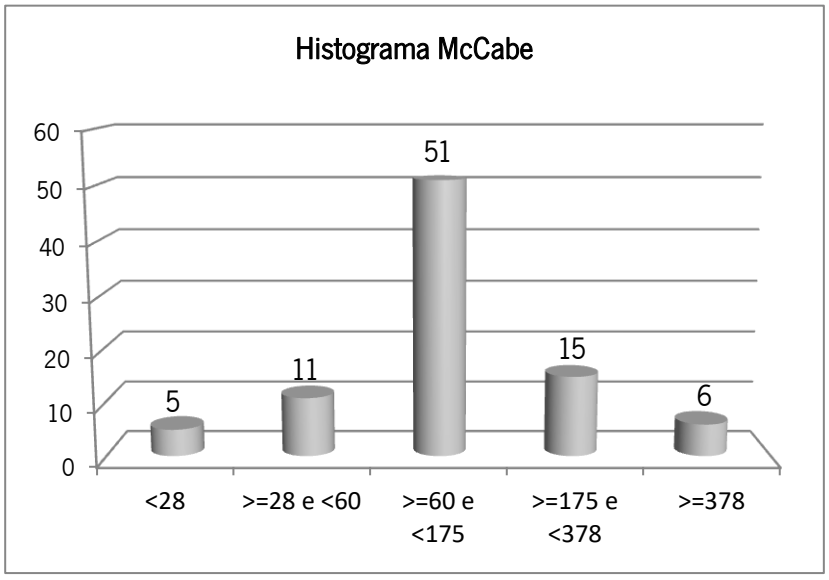


Figura 20 - Histograma de Complexidade de McCabe

Como amostra dos programas realizados entre os 88 desenvolvidos pelos alunos de 2011 a 2014, selecionei 5 programas que estão na região da Média Baixa (ver Figura 17), considerando simultaneamente os critérios de extensão e complexidade. Isto quer dizer que os 5 programas das amostras baixas estão entre 119 e 254 linhas de códicos (Ver Figura 19) e possuem complexidade de McCabe entre 28 e 60 (ver Figura 20). Selecionei, também, outros 5 programas que estão na região da Média Alta(ver Figura 17), considerando simultaneamente os critérios de extensão e complexidade. Isto quer dizer que os 5 programas das amostras alta

estão entre 723 e 1537 linhas de códicos (Ver Figura 19) e possuem complexidade de McCabe entre 175 e 378 (ver Figura 20). A amostra tem, portanto, 10 programas.

5 Apresentação dos dados e sua discussão

Nesta pesquisa investigo como o ensino de programação de computadores pode contribuir para melhorar a formação dos futuros professores de matemática, acreditando que eles devam praticar as competências do pensamento computacional, em especial a autonomia e criatividade. Para isso, relato o trabalho que fiz desde 2010, quando comecei a lecionar a disciplina de Introdução a Lógica de Programação na Universidade Federal do Recôncavo da Bahia, até ao ano de 2014. A realidade que encontrei revelava um ensino de programação limitado à memorização de técnicas para resolução de problemas que não faziam sentido para os alunos. Realizei algumas tentativas para melhorar minhas aulas com o objetivo de formar alunos mais criativos e autônomos e, no final desses anos, acredito que tenha conseguido melhorias no processo de aprendizagem da disciplina. Ao tentar modificar esta realidade com uma intervenção consciente, constato que a pesquisa está alinhada com o paradigma sóciocrítico (Coutinho, 2013). Ao realizar a pesquisa a partir das atividades experimentadas em sala de aula, constato que estou utilizando uma metodologia indutiva para descobrir contribuições para o processo de aprendizagem (Coutinho, 2013).

Na autobiografia mostro as diversas intervenções que levaram ao amadurecimento das práticas adotadas. Para avaliar os resultados alcançados pelo processo descrito na autobiografia, utilizo a pontuação relativa ao pensamento computacional estabelecida pela ferramenta Dr. Scratch.

Uma vez estabelecido o processo utilizado nas aulas da disciplina, faço uma análise documental procurando evidências de qualidade nos processos para atingir um ensino que desenvolva a autonomia e a criatividade, ultrapassando a simples memorização de técnicas. Para avaliar as evidências da análise documental, utilizo a taxonomia de Bloom como técnica de avaliação da qualidade do processo estabelecido.

5.1 Propostas de trabalho apresentadas aos alunos

Sob este tema pretendo mostrar como evoluiu a minha práxis de ensino, podendo servir de referência para outros professores. Modifiquei o processo de aprendizagem e o método de avaliação, tentando influenciar o comportamento dos alunos no sentido de serem mais criativos

e autônomos. Ao descrever a experiência vivida como professor da disciplina de Introdução a Lógica de Programação, contextualizando as tentativas, erros e acertos que descrevem a construção do processo pedagógico, procuro dar uma perspectiva da evolução do processo utilizado na disciplina apresentando os dados em ciclos anuais, iniciando em 2010 e finalizando em 2014.

Em 2010 ainda não utilizava o Scratch como ambiente e linguagem de desenvolvimento de software. Sendo assim, 2010 pode ser considerado como a situação inicial que me levou a fazer a pesquisa-ação a partir de 2011.

5.1.1 Ciclo letivo de 2010

Em 2010 comecei a lecionar a disciplina de Introdução a Lógica de Programação na UFRB em Amargosa-BA. Em conversa com a professora que me precedeu e atendendo às referências bibliográficas, notei que se tratava de um ensino de programação estruturada com linguagem de 3.ª geração. A Professora anterior utilizava linguagem Pascal no final do curso e seguia a sequência preconizada nos livros tradicionais, isto é, ensino de fluxogramas, de uma linguagem de computador hipotética conhecida como Português Estruturado (ou Portugol). Esta linguagem hipotética encontrada nos livros (Forbellone & Eberspächer, 1993; Manzano & Oliveira, 2000) parece uma tradução da linguagem Pascal para o português. Os conteúdos eram passados conforme aparecem nos livros didáticos. Conceito de algoritmo, tipos de variáveis, estruturas de programação (sequência de comandos, condicionais e loops) eram passados um por vez a cada aula. Os exercícios eram realizados com papel e lápis utilizando o Português Estruturado. O contato com o computador se dava no final do semestre quando era solicitado um programa em Pascal.

Eu vinha de um histórico de desenvolvimento de software profissional com alta produtividade e recursos diversos de programação. Todo o contexto da disciplina me parecia anacrônico, dando a impressão de que a disciplina havia parado nos anos 80. No entanto, respeitando a minha pouca experiência como professor da disciplina, não introduzi muitas modificações tecnológicas. Concentrei minhas energias na metodologia de ensino, porque eu acreditava que o aluno deveria ser capaz de programar, superando as dificuldades necessárias a este processo. Para mim, o ensino de uma linguagem hipotética (Portugol) não fazia sentido, uma vez que era necessário ensinar uma linguagem real (Pascal ou C). Fluxograma era uma

ferramenta que eu já não utilizava desde os anos 80. Hoje em dia os documentos sugeridos pela Unified Model Language (UML) são os utilizados pelo mercado (Casos de Uso, Diagrama de Classes, Diagrama de Sequências etc.). Por tudo isso, decidi ensinar a linguagem real indo diretamente a ela e com um enfoque prático, levando os alunos frequentemente ao laboratório de informática.

Dividi as aulas semanais em duas teóricas e duas práticas. Nas teóricas eu explicava os algoritmos e os programas propostos e nas aulas práticas propunha programas a serem desenvolvidos com o auxílio de procedimentos, isto é, roteiros, instruindo detalhadamente o que o aluno deveria programar. Adotei a linguagem C, por estar mais familiarizado com ela do que com a linguagem Pascal. Também por achar que deveria utilizar *software* livre, para que os alunos pudessem instalar o ambiente de desenvolvimento no seu próprio computador, adotei o Dev C++, um ambiente de desenvolvimento de programação integrado, baseado no projeto GNU de software livre (Senne, 2006). Para a avaliação na disciplina construí duas provas com perguntas sobre comandos da linguagem C, e solicitando algoritmos semelhantes ou iguais às que tínhamos visto em classe. O modelo de aula seguindo o livro, isto é, uma aula teórica apenas sobre variáveis, outra apenas sobre condicionais e assim por diante, de forma fragmentada, não me pareceu uma boa abordagem, porque, ao programar, todos estes conceitos vêm ao mesmo tempo. Por essa razão, resolvi iniciar por introduzir programas simples e ir aumentando a complexidade, abordando os conceitos à medida que eram utilizados, estabelecendo um paralelo entre aulas práticas e teóricas.

Nas aulas em laboratório, era evidente o tempo gasto com a correção dos erros de sintaxe. Frequentemente o ponto-e-vírgula, obrigatório como demarcador de comando em C, era esquecido. Erros estruturais, como o esquecimento de um fecha-chaves impediam os alunos de continuar sem a minha intervenção. As mensagens de erro em inglês e pouco esclarecedoras dificultavam a depuração (*debug*) dos programas. Os alunos tinham dificuldades para criar novos algoritmos. Por isso, nas provas, havia algoritmos já estudados ou com pequena variação de um algoritmo já conhecido.

5.1.1.1 Considerações sobre o ciclo letivo de 2010

As aulas aconteciam em dois encontros semanais de duas horas. No primeiro encontro eu mostrava a proposta do programa que iríamos desenvolver, assim como a lógica e a técnica

envolvida. No segundo encontro íamos ao laboratório para implementar a proposta que estava detalhada do roteiro da aula. Notei que os alunos se interessavam mais pela parte teórica do que pela parte prática. No laboratório, a correção dos erros de sintaxe ocupava a maior parte do tempo. Os alunos se valiam da memorização e repetição para conseguir resultados modestos. Eles não eram capazes de propor um programa e codificá-lo por completo. Com muito esforço conseguiam fazer pequenas variações do que já estava codificado e funcionando. Eu achava que, se eles tivessem um ambiente para produzir em uma interface gráfica, o interesse poderia aumentar.

5.1.2 Ciclo letivo de 2011

Em 2011 iniciei utilizando o Scratch. Com o Scratch não se tinha mais o desconforto do uso de telas pretas, pouco atraentes, do tipo MS-DOS. Também pude evitar problemas que não entusiasmavam os alunos como “ache os números primos até 100” ou “ache o valor máximo em uma matriz”. Com o Scratch foi possível o uso de multimídia com uma interface gráfica que possibilitava o uso de imagens e sons.

O Scratch me foi apresentado por um amigo, professor da Universidade Federal de São Carlos. Rapidamente percebi o potencial do ambiente Scratch e passei a pensar nas aulas de Introdução a Lógica de Programação com esta ferramenta. Fiz alguns programas iniciais propostos pelo site http://scratch.mit.edu/. Escolhi um conjunto básico de comandos e elaborei programas-exemplo: “Hello World”, “Dancing Queen”, “Conversão Milhas Quilômetros”, “Andamento do ciclista”, “Carro”, “Moeda”, “Labirinto”, “Caça Fantasma”. Cada programa tinha uma intenção de ensino de um conteúdo específico. Em seguida elaborei roteiros que os alunos pudessem seguir no laboratório.

Dividi o curso de programação com Scratch em três etapas: Roteiros, Prova e Projeto.

– *Etapa Roteiros*

Esta etapa está reservada para que o aluno adquira os conhecimentos básicos da linguagem Scratch e ganhe competências de utilização do ambiente de programação. Continuo com a estratégia de duas horas de aula teórica e, em outro dia, duas horas de aula de prática. Na aula teórica eu explicava o programa que faríamos em nosso próximo encontro, abordando os conceitos necessários para entender o código. Na aula prática os alunos me chamavam a cada dificuldade e havia reclamações por eu não atender a todos. Por não conseguir dar conta

de todos, acabei por solicitar para alguns que tinham mais facilidade que me ajudassem, esclarecendo as dúvidas dos colegas. No final da etapa básica, os alunos se ajudavam uns aos outros e eu já conseguia dar conta das dúvidas que chegavam até mim.

Os roteiros funcionavam bem. Existia um crescente de dificuldades, por exemplo, no primeiro o gato  (personagem predefinida do Scratch) dizia “Hello, World” e depois mostrava com detalhes como salvar o programa. No segundo roteiro mostrava com detalhes como fazer um contador e, para salvar não havia mais o processo detalhado: dizia simplesmente “salve o programa”. No terceiro roteiro usámos uma função linear para deslocar o gato, utilizando um contador como variável independente. Para fazer o contador não havia instruções detalhadas, dizia simplesmente “crie um contador de 0 a 100”.

Os roteiros apresentavam os comandos já sendo utilizados e as explicações de como funcionava cada comando já vinham em um contexto. Considerei este processo mais interessante do que fazer aulas expositivas sobre como funciona cada comando. O Scratch tem uma organização dos comandos identificados por cores. Por exemplo, comandos de movimento do objeto no plano cartesiano estão em azul e são acessados ao clicar no menu Movimento; comandos de controle são da cor laranja e são acessados ao clicar no menu Controle (Figura 21). Esta informação não foi ensinada em nenhuma aula teórica, mas os alunos aprenderam observando e tirando suas próprias conclusões ao fazer os roteiros.

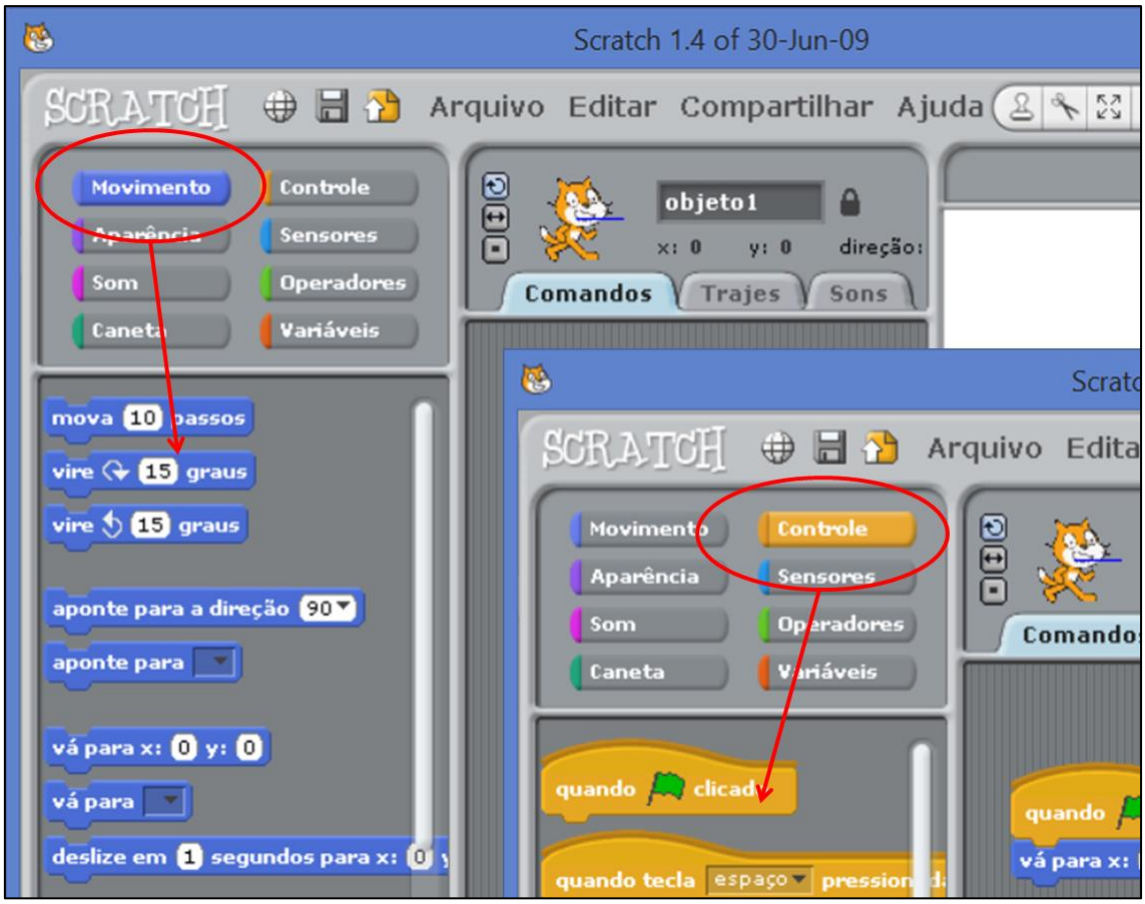


Figura 21- Acesso as Classes de Comandos via Menu

– *Etapa Prova*

A prova é o tradicional teste escrito, sem consulta, para avaliar individualmente a proficiência dos alunos. Neste ano de 2011 fiz uma prova bem parecida com as de 2010, isto é, com algoritmos iguais ou muito parecidos com os que vimos em sala de aula, para ser escrito na prova.

As notas eram baixas e passei a me questionar sobre o porquê dessas notas. Pensei que seria melhor pedir para fazer programas no computador, mas não havia computadores suficientes no laboratório de informática. Fiquei com isso na cabeça, mas lápis e papel não eram bons para fazer programas. O retorno que o computador dá é parte do processo de fazer programas para computador. Com lápis e papel não existia o ciclo descrição-execução-reflexão-depuração (Maltempi & Valente, 2000).

– *Etapa Projeto*

Já no início de 2010, quando iniciei as aulas de Lógica de Programação, os alunos veteranos contavam que o projeto de *software* era onde aprendiam mais. Quando meu colega, professor da Universidade Federal de São Carlos (UFSCar), me disse que utilizava o Scratch, ele me mostrou alguns projetos desenvolvidos por alunos que me deixaram impressionado. Ingenuamente, achei que meus alunos não atingiriam tal desempenho em programação. Gostei da ideia de fazer um projeto final para que eles pudessem mostrar o que aprenderam. O projeto também seria outra forma de avaliar, evitando a prova escrita, em relação à qual eu tinha certas restrições. Mostrei alguns dos projetos desenvolvidos pelos alunos da UFSCar para que meus alunos tivessem uma ideia do que queria que eles fizessem.

Para considerar o projeto final como elemento de avaliação, pensei que deveria orientá-los a planejar antes de codificar. Os antigos fluxogramas ou os modernos diagramas da UML não me pareceram adequados. A solução que encontrei foi a utilização de um *storyboard*, porque isso daria o número de cenas e uma ideia da complexidade de cada uma delas. Seria possível identificar os objetos a serem codificados em cada cena. O tema seria livre, mas sugeri que era interessante que escolhessem um tema de física ou de matemática do ensino básico como inspiração para o projeto. Separei cinco semanas de aulas para o projeto, sendo uma para o *storyboard*, três para a programação com Scratch e mais uma para as apresentações dos projetos, totalizando 20 horas de aula. Havia uma orientação de que o projeto deveria interagir com o usuário, o que é uma característica dos programas, não podia ser uma apresentação que poderia ser realizada com o Power Point. Nesta etapa, os alunos elegem as atividades a serem desenvolvidas para chegar ao resultado desejado.

Na criação do *storyboard*, observei que os alunos tinham dificuldades em começar. Várias vezes me perguntaram “professor, o que é pra fazer no projeto final?”. Eu respondia, “o que você quiser”. Esta dificuldade apareceu nos dois encontros semanais, mas depois de elaborar um *storyboard*, eles se sentiram mais confiantes.

Nas aulas dedicadas à programação aconteceu de alguns grupos chegarem e pedirem por procedimentos ou perguntarem o que era para fazer. Neste instante eu relembrava que nesta etapa era para eles realizarem a programação dos projetos, e que, se encontrassem dificuldades, eu poderia ajudar. Na primeira semana, alguns grupos ficaram perdidos, sem saber o que fazer. Mas nas últimas semanas havia uma intensa movimentação na sala. Os alunos resolviam os problemas entre eles e minhas intervenções foram poucas.

As apresentações transcorreram muito bem. Alguns projetos eram bem simples, com perguntas esperando respostas pelo teclado e retornando certo ou errado. Outros eram mais elaborados, com temas diversos. Alguns mostram a cultura e personalidade dos alunos. Nesse aspecto destacamos os projetos "Vida de Estudante" e "As leis de Newton". Em "Vida de Estudante" são apresentados conteúdos de matemática financeira, discutindo os gastos e o modo de vida dos estudantes da UFRB. Em "As Leis de Newton", uma aluna do grupo passa a ser personagem da história, posando e cedendo imagens para a realização do projeto. Deste modo, o projeto em Scratch serviu como uma forma de expressão para este grupo de alunos.

Uma reclamação desta primeira turma a utilizar o Scratch foi que os projetos eram um salto muito grande em termos de autonomia. Na minha avaliação, os alunos não estão acostumados a exercerem esta autonomia. O *storyboard* serviu como planejamento e como controle de prazos, fazendo com que os alunos refletissem sobre o gerenciamento do seu trabalho.

5.1.2.1 Considerações sobre o ciclo letivo de 2011

O uso do Scratch se mostrou acertado para o ensino de programação. O fato de os alunos conseguirem realizar seus próprios projetos é o que mais me motivou. Os alunos conseguiam uma autonomia que não existia com as linguagens Pascal ou C. A composição dos programas, encaixando os comandos e as estruturas condicionais e de repetição, veio resolver os problemas de sintaxe que estressavam os alunos. Sem estresse da retirada dos erros de sintaxe, os alunos podiam se concentrar na lógica da programação. O entusiasmo dos alunos e os projetos finais indicavam um ganho significativo na aprendizagem e na empatia dos alunos com relação à programação de computadores, quando comparado com 2010.

Os roteiros, já utilizados em 2010, se mostraram claramente acertados em 2011, uma vez que os problemas de sintaxe anteriores comprometiam a avaliação. Outra constatação é a de que os alunos aprenderam coisas que não precisaram ser ensinadas, como os detalhes da ferramenta Scratch, a divisão das classes de comandos ou mesmo a explicação de cada comando de forma exaustiva. Isso confirma o que nos fala Sugata Mitra que, com um mínimo de intervenção, acontece a aprendizagem em um ambiente propício (Mitra, 2000).

Realmente, eu não poderia atender todas as dúvidas de todos os alunos, mas, na medida em que eles passaram a perguntar aos colegas, as coisas fluíram melhor. Acredito que

passaram a entender que não dependiam apenas do professor e que o colega poderia ser uma ajuda eficaz para o seu problema localizado. Deste modo o ensino passou a ser mais socializado e o ambiente de aprendizagem melhorou uma vez que os laços sociais foram reforçados, como preconizam as teorias sócio-interacionistas.

As provas não me deixaram satisfeito. Eu acreditava que elas não refletiam o conhecimento que os alunos mostravam nos projetos e estava convicto de que a solução seria ter um computador por aluno e as provas deveriam ser realizadas no computador, mas isso seria inviável pelas condições de trabalho que se apresentavam.

Os projetos foram o ponto alto da disciplina. O *storyboard* se mostrou acertado como forma de planejar e gerou discussões sobre prazos e a carga de trabalho que eles mesmos estavam se propondo. Com estas discussões acredito ter ganhado a confiança dos alunos, quando os levava a pensar no prazo e na carga de trabalho que estavam se propondo a fazer.

Os projetos foram feitos em grupos de quatro pessoas e percebi que alguns, poucos, não se interessavam nem contribuíam para o projeto, valendo-se das anotações dos demais. Uma constatação importante foi a reclamação dos alunos de que o projeto era muito mais difícil do que os roteiros e que no princípio ficaram assustados. Em conversa, no último dia de aula, quando eu e os alunos nos reunimos para avaliar o curso, mencionei que talvez devesse propor um projeto mais simples antes desse projeto final. Os alunos concordaram que assim seria melhor.

5.1.3 Ciclo letivo de 2012

Em 2012 mantive o mesmo esquema de intervenção do ano anterior, apenas acrescentei um pequeno projeto de animação antes do projeto final, procurando responder às queixas dos estudantes. Deste modo, a etapa de projeto foi subdividida entre um pequeno e um grande projeto.

- *Etapa Roteiros*

Na etapa básica acrescentei algumas aulas para reforçar conceitos e evitar os enganos mais recorrentes detectados em 2011:

  - Confusão das instruções do Scratch versão 1.4 entre “mude [variável] *por* [constante]” e “mude [variável] *para* [constante]”.

- Reforçar a conceito de contador, com uma aula específica na qual os alunos faziam contadores crescentes e decrescentes.
- Passar mais tempo trabalhando em dois programas muito básicos (Contador e Andamento do Ciclista) que mostravam as estruturas loop e condicional, respectivamente.

Nas aulas de laboratório já solicitei que pedissem ajuda aos colegas e não apenas a mim. Isso devido à experiência do ano anterior. Considerando a teoria sócio-histórica, reparei que os roteiros colocam os alunos em situações nas quais eles vão aprendendo na tentativa de acertar. O que um aluno não é capaz de fazer sozinho, consegue com uma ajuda de seus colegas e do professor, para depois conseguir fazer sozinho. Em outras palavras, os roteiros colocam os alunos em zonas de desenvolvimento proximal para que ele se desloque da zona de desenvolvimento real para a zona de desenvolvimento pretendida. Como as zonas de desenvolvimento proximal são diferentes em cada aluno, o aluno menos competente é auxiliado por outro mais competente naquele aspecto em particular.

Nos roteiros eu incentivava os alunos a modificarem os programas. Com este incentivo, surgiram personagens do universo dos alunos, como os personagens da Turma da Mônica e de Os Simpsons, entre outros, sendo o Chaves e o Bob Esponja os mais frequentes. Para exemplificar, cito um programa de labirinto modificado por uma aluna que criou 10 fases para levar o gato (visto de cima: do canto inferior esquerdo, para o círculo preto, no canto superior direito), com uma estética própria. Na imagem seguinte coloquei quatro dessas fases para ilustrar algumas dessas fases (ver Figura 22).

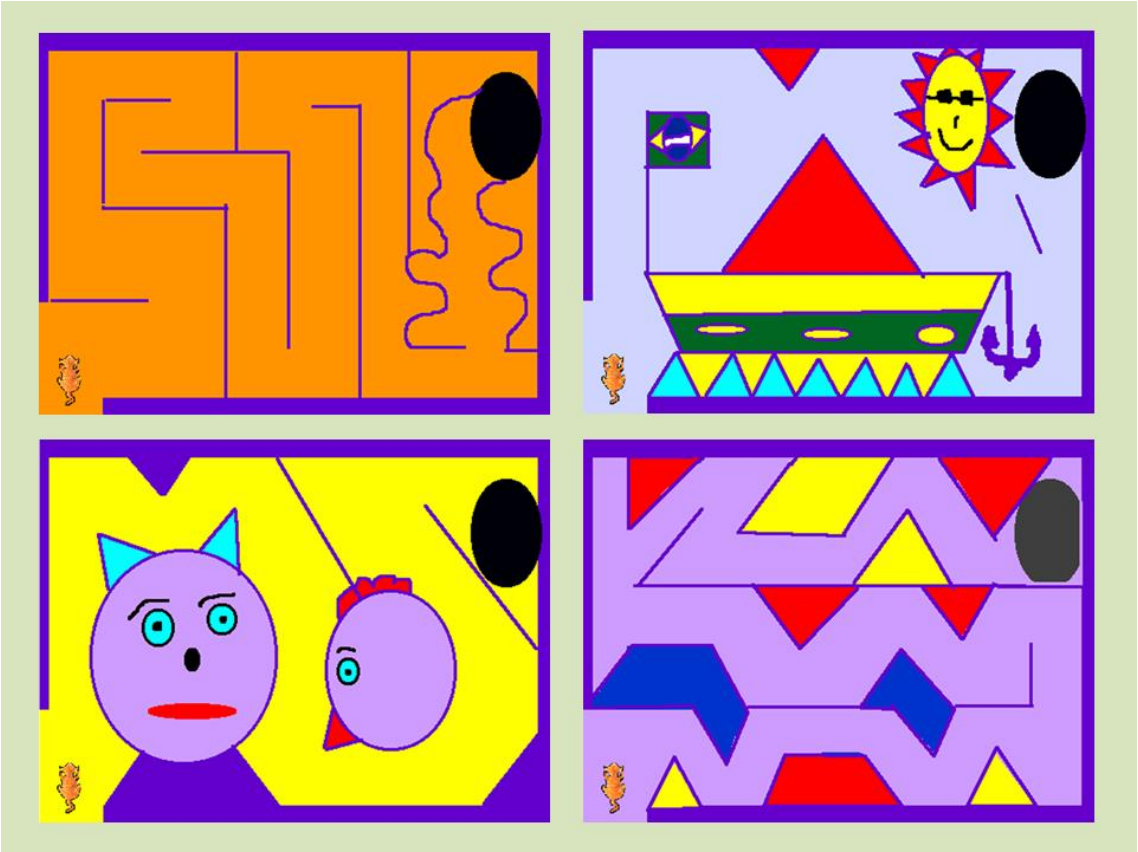


Figura 22 - Exemplo de uma Estética Adotada por uma Aluna

– *Etapa Prova*

Conversando com um colega do curso de matemática, ele me falou de uma prática que adotou em sala de aula. Ele pedia para os alunos escreverem as estratégias que adotaram na resolução dos problemas matemáticos. Fiquei pensando que seria uma opção para a prova. Também tive contato com textos de Marcelo Borba como "Coletivos seres-humanos-com-mídias e a produção de Matemática" (Borba, 2002), "Humans with Media: a performance collective in the classroom?" (Borba, 2007), "Humans-with-media and continuing education for mathematics teachers in online environments."(Borba, 2012). Borba sustenta que Seres-Humanos com suas mídias são a unidade epistemológica do aprendizado. Baseado nas teorias de Lev Vygotsky e Pierre Lévy, Borba (Borba, 2007, 2012). Borba considera que um coletivo de pessoas pensa de forma diferente de acordo com a mídia que usa para se expressar. Isso me fez pensar que mídia de papel e lápis não era adequada para fazer programas, mas eu poderia adaptar a prova à mídia de papel e lápis pedindo para que os alunos descrevessem o funcionamento dos programas, algo semelhante ao que fazia meu colega professor de matemática.

Deixei uma aula na qual eles descreveram um programa em duplas. Na semana seguinte comentei os resultados. Esclareci os enganos mais comuns e passei algumas recomendações sobre como deveriam descrever o programa. Uma aula depois apliquei a prova. Agora ela tinha dois pequenos programas onde se pedia para os alunos explicarem o funcionamento. Eles deveriam relacionar os comandos com os respetivos efeitos.

Depois da prova percebi que as notas continuavam baixas, mas dessa vez eu conseguia detectar erros conceituais. Quando os alunos referem “o loop se” ou “a variável faz”, eu entendo que estão confundindo o condicional *se* com *loop* e que não perceberam que uma variável só armazena dados, e os comandos fazem as coisas acontecerem. Notei também as dificuldades de expressão na língua escrita. Alguns utilizam frases sem sentido e outros faziam uma cópia dissimulada do código (Figura 23).

Programa	Cópia Dissimulada
	Inicia com quando bandeira clicada Em seguida muda Contador para 3 Repete até Contador > 15 Depois diga Contador por 0,5 segundos Muda contador por 2

Figura 23 - Exemplo de cópia dissimulada do código Scratch 1.4

– *Etapa Projetos*

Por conta das sugestões dos alunos no ano anterior, resolvi solicitar um projeto de animação. Uma história deveria ser contada, movimentando os atores com imagens e sons. Minha intenção foi criar um pequeno projeto antes do grande projeto final da disciplina. A proposta consistia em organizarem-se em duplas e que cada dupla realizaria uma parte de um projeto maior da classe. Imaginei que a tarefa poderia ser uma sequência como aquelas armadilhas de desenho animado nas quais a bola bate no pino, que dispara uma mola, que lança uma faca, que corta uma corda, que cai uma gaiola para prender a presa. Para isso passei alguns vídeos com estes temas, inclusive a abertura do programa Ra-Tim-Bum, programa infantil brasileiro, produzido pela TV Cultura (https://www.youtube.com/watch?v=TpcbIxLbDGk), na tentativa de despertar o interesse com uma lembrança da infância.

Este projeto foi desenvolvido em três encontros, sendo um para elaboração do *storyboard* e dois para codificação do programa. As duplas que não estavam com o *storyboard* definido no segundo encontro ficaram sem saber o que tinham que codificar. Nestes casos, recomendei que definissem o *storyboard*, finalizando a etapa que iniciámos no encontro passado.

A proposta de realização de uma animação resumia-se a uma bola que entra em cena pelo lado esquerdo, é jogada de um lado a outro pelo menos três vezes e depois sai pelo lado direito. A altura de entrada e saída foi padronizada para dar um efeito de continuidade ao juntar as animações.

Para a entrega das animações, promovi a elaboração de um vídeo em sala de aula. Forneci uma máquina fotográfica aos alunos e pedi que tirassem fotos para colocar na primeira parte do vídeo. Enquanto isso eu converti as animações em vídeos e pedi para um aluno voluntário montar o vídeo e colocar uma música de fundo. Ao final da aula produzimos um vídeo com fotos no início seguidas das animações desenvolvidas. A exposição do vídeo na rede social gerou repercussão que ultrapassou os limites da sala de aula. Um dos vídeos resultantes pode ser acessado em https://youtu.be/OwSpJtr1GJI

O projeto final de 2012 seria feito utilizando uma semana de *storyboard*, duas de codificação e mais uma para apresentação. Os projetos transcorreram com mais fluidez, sem aquele impasse de não saber como começar o *storyboard*. Com a aprendizagem da experiência do semestre anterior eu também pude orientar melhor. Para definir a primeira versão do *storyboard*, recomendei a seguinte estratégia:

- Escolher o tema;
- Discutir em linhas gerais como será a história;
- Desenvolver um *storyboard* com um quadro para cada cena.

Passei também a solicitar que, a cada semana, os alunos me entregassem uma folha descrevendo a situação atual do projeto e como eles esperavam que estivesse na semana seguinte. Na semana seguinte, eu apresentava a folha da semana anterior para que pudessem ver se suas expectativas se tinham realizado.

5.1.3.1 Considerações sobre o ciclo letivo de 2012

A solicitação feita para que os alunos modificassem os programas Scratch gerou alguns resultados interessantes, demostrando envolvimento e criatividade. O programa personalizado cria um laço afetivo com o objeto criado e curiosidade por parte dos colegas.

A prova explicativa se mostrou mais adequada e reveladora do que as tradicionais. Adequada, por estar de acordo com a tecnologia de papel e lápis e com a explicação extensiva do programa. Reveladora, porque mostrou os erros conceituais e deixou clara a dificuldade dos alunos em se expressarem através da escrita. Mesmo que as classificações não sejam melhores, o fato de cobrar o entendimento e a forma de expressão é, na minha opinião, qualitativamente melhor que a memorização e reprodução de algoritmos. Elieser de Jesus e Alice Raabe também repararam neste aspecto e referem que

> o problema em se utilizar questões discursivas onde, por exemplo, o aluno descreve o funcionamento de um trecho de código é que o desempenho pode ser ruim não por falta de entendimento sobre o funcionamento de programas, mas sim pela própria dificuldade de escrever. (E. A. Jesus & Raabe, 2009, p. 5)

Tal como Jesus e Raabe, percebo que os alunos têm dificuldades de se expressar por escrito. Este é um desafio que nós, professores de matemática do Centro de Formação de Professores da UFRB, percebemos e enfrentamos junto com eles até a monografia final, na conclusão do curso. Porém, ao contrário daqueles autores, eu creio que se deve manter o desafio da escrita para aprender a se expressar pela escrita. Tal como para aprender a programação, a ser professor e a andar de bicicleta, a prática é a parte mais importante do processo. Acredito que estarei contribuindo melhor para a formação dos meus alunos, tanto no curso de matemática como para a sua profissão de professor, se o desafio da escrita for mantido. Por isso, não recorrerei a questões de múltipla escolha, contrariando o que sugerem estes autores Lister (2000) e Jesus e Raabe (2009).

O projeto de animação se mostrou adequado à introdução da realização de projetos e administração do próprio trabalho. O espaço de tempo, relativamente curto, de uma semana para execução destacou a necessidade de iniciar a codificação com um objetivo claro e revelou a importância da utilização do *storyboard* no processo. Acredito que estas lições aprendidas com a animação, fizeram com que os alunos iniciassem o projeto final de 2012 com mais confiança do que em 2011. Notei também que as técnicas que expliquei no decorrer das aulas se faziam necessárias no projeto, mas minhas explicações, apesar de mais superficiais do que nas aulas, parecem ser mais bem compreendidas pelos alunos. Não sei por que isso acontece, mas fico imaginando que durante as aulas as técnicas servem para solucionar problemas que são meus, mas, quando chega a vez de realizar o projeto, os alunos precisam das técnicas para resolverem os problemas deles, e aí, de algum modo, o interesse aumenta e a compreensão fica mais fácil.

5.1.4 Ciclo letivo de 2013

Ainda em 2012, promovi um evento cultural sobre *videogames*, com a intenção de mostrar que existe uma cultura relacionada a estes jogos. O entusiasmo dos alunos me surpreendeu, e eu passei a dar mais importância aos jogos como atrativo para o ensino. Vi a oportunidade de passar roteiros com programas mais complexos. Com isso criei alguns jogos simplificados de “Batalha de Cartas”, “Wumpus” e “Space Invaders” e acrescentei a etapa

básica do curso. Pouca coisa mais se modificou com relação a 2012, por isso não entrarei em muitos detalhes. Reduzi os grupos do projeto de quatro para três elementos, na tentativa de chegar mais próximo a uma avaliação individual.

5.1.4.1 Considerações sobre o ciclo letivo de 2013

Ao final, percebi que realmente os alunos conseguiram realizar roteiros mais complexos, porém, eles ficavam presos a realizar, sem tempo para personalizar, sacrificando a parte criativa do processo. Não arrisquei colocar em duplas porque não sabia se iria conseguir corrigir todos os projetos a contento. Na avaliação dos projetos de 2013, consegui um código em Java que, depois de modificado, obtinha o número de comandos utilizados em cada projeto Scratch. A ferramenta não é amigável e eu tive que modificar o código fonte para cada programa analisado, mas me ajudou a perceber que as notas subjetivas que eu atribuía tinham uma certa coerência com o número de comandos utilizados. Esta medida pode ser considerada como LOC.

5.1.5 Ciclo letivo de 2014

Em 2014 algumas alterações aconteceram. A maioria delas devido a leituras sugeridas pelos meus orientadores na Universidade do Minho e também pela opção de captar dados mais sistematicamente para a elaboração do meu trabalho.

Adotei o conceito de Sala de Aula Invertida (Flipped Classroom), que basicamente coloca as aulas expositivas em vídeos para serem assistidas em casa e o que seria a lição de casa, dever, ser realizada na classe com assistência do professor e dos colegas.

Continuei achando que a cultura do *videogame* seria uma maneira de motivar os alunos. Só que, desta vez, ao invés de propor roteiros mais complexos, coloquei alguns mecanismos de jogos simples que deveriam ser aproveitados no projeto final. Acrescentei no plano de aulas os mecanismos de jogos que seriam apresentados. Assim, a etapa de projeto ficou subdividida em Animação, Mecanismos de Jogos e Projeto Final.

Logo no início das aulas, pedi para os alunos explorarem um site com jogos do Atari (http://www.jogosdeatari.com.br). Fiz isso imaginando que os mecanismos são possíveis de programar com Scratch e que isso despertaria as possibilidades de realização do projeto final.

- *Etapa Roteiros*

Na etapa básica, continuei reforçando o ensino do contador, da diferença entre "mude para" e "mude por" no Scratch 1.4 e os programas básicos do "contador" e "andamento do ciclista".

– *Prova*

Para treinar os alunos a descrever programas, pensei que poderiam descrever os mecanismos adaptados por eles mesmos. Imaginei que em um programa adaptado eles já teriam reflexões a respeito do seu funcionamento. No entanto, alguns tiveram dificuldades em adaptar, e muitos estavam com *bugs* que comprometiam o seu funcionamento. Por isso, abandonei a ideia de descrever a adaptação de cada dupla e estabeleci quatro programas que deveriam ser descritos. Os dois primeiros foram descritos em sala de aula e outros dois ficaram para serem descritos como tarefa de casa. No nosso encontro seguinte, as explicações foram trocadas e cada um avaliou as explicações de dois outros colegas. Em um terceiro encontro os colegas avaliadores deram retorno aos avaliados. Depois dessas atividades, apliquei a prova no encontro seguinte.

– *Etapa Projetos*

Com relação ao projeto de Animação, em 2011 e 2012 solicitei animações de uma bola que é rebatida em cada cena e continua na cena seguinte, desenvolvida por outra dupla de alunos. Depois de tentar, sem sucesso, ter uma outra boa ideia, resolvi pedir à classe a sugestão de um tema para uma animação comum. A classe elegeu um tema para animação, através de votação. Deste modo, ao juntar as animações de cada dupla formou-se uma unidade, um fio condutor que justificava a junção das animações compondo o vídeo da turma.

Entre a Animação e o projeto final me dediquei a ensinar alguns mecanismos de jogos. Mostrei alguns mecanismos de jogos, como atirar em objetos que se movem na tela ou deslocar um objeto por um labirinto, entre outros mecanismos. Depois solicitei que adaptassem um deles para que se dedicassem a ele com mais afinco. Esta atividade poderia ser aproveitada para o projeto final. Por isso, o projeto final foi conduzido como uma evolução da Animação no qual deveria haver uma narrativa intercalada com mecanismos de jogos. Continuei acompanhando a evolução a cada semana, mas, desta vez, deixei clara a expectativa que tinha com relação às três semanas que se seguiriam para o desenvolvimento do programa:

- Na primeira semana o *storyboard* deveria estar pronto,
- Na segunda semana pelo menos 50% das cenas deveriam estar prontas,

– Na terceira semana deveria rever-se o projeto para conciliar o que se pretendia realizar com o prazo que restava.

Com a Animação os alunos aprenderam a fazer o *storyboard* e não houve dúvidas para o *storyboard* do projeto. Eles apresentaram dificuldades para adaptar os mecanismos às cenas do projeto final. Os projetos foram realizados em duplas. Com a experiência bem sucedida dos projetos em trios no ano anterior e a descoberta da métrica de *software* de linhas de código como parâmetro de coerência para a avaliação, fiquei confiante em estabelecer projetos em duplas.

5.1.5.1 Considerações sobre o ciclo letivo 2014

Adotar os vídeos para minhas aulas expositivas, seguindo a ideia da sala de aula invertida, foi um processo fácil para mim, uma vez que muitas das aulas já eram essencialmente práticas e a parte expositiva era baseada nos roteiros realizados. A dificuldade foi com a gravação de vídeos quase toda semana. Com a continuação dessa prática acredito que haverá o reaproveitamento de alguns vídeos e a experiência me fez criar um processo razoavelmente eficiente para a produção dos vídeos. Os alunos gostaram da experiência assinalando como vantagem poderem rever as aulas quando tinham dúvidas. Para motivar os alunos a assistirem os vídeos eu colocava um pequeno questionário com 3 a 5 perguntas diretamente relacionadas ao vídeo, no ambiente virtual de aprendizagem, com o prazo de uma semana para ser respondido.

No projeto final, a narrativa não precisa ser sequencial, podendo uma cena se repetir ou não aparecer, dependendo do fluxo tomado a partir dos resultados no jogo e das interações com o usuário. A interação com o usuário, a narrativa com fluxos alternativos e a liberdade de escolha do tema colocam o aluno frente a desafios maiores do que a animação.

Os *storyboard* ficam desatualizados ao final do projeto. Em um contexto profissional de desenvolvimento de *software* isso seria um problema, porque ter documentação que não está atualizada compromete a futura manutenção. Mas, no nosso contexto de sala de aula, isso não é um problema, porque não será dada a manutenção desse software no futuro e o mais importante é que o aluno fique livre para modificar o projeto, aperfeiçoando e se divertindo com o trabalho e com o consequente aprendizado.

5.2 Resultados da análise do Processo de Aprendizagem

Com a experiência ministrando a disciplina de Introdução à Lógica de Programação, estabeleci um processo baseado em atividades nomeadas por Roteiros, Prova e Projetos, a partir de observação, planejamento, intervenção e avaliação em ciclos anuais de 2011 a 2014. Este processo foi estabelecido de forma subjetiva a partir das observações e impressões obtidas em sala de aula.

Neste item pretendo avaliar os resultados de forma menos subjetiva, de duas maneiras. A primeira será a avaliação do pensamento computacional a partir de uma amostra de 10 programas realizados nestes quatro anos de investigação. A segunda forma de avaliação será a análise dos documentos que propõem as atividades de Roteiros, Provas e Projetos, avaliando o valor cognitivo dessas atividades utilizando como referência a taxonomia de Bloom.

5.2.1 Indicadores de Pensamento Computacional

Para avaliar a produção dos alunos tendo o pensamento computacional como base, utilizarei a ferramenta Dr.Scratch. Esta ferramenta calcula um valor que varia de 0 a 21 considerando as evidências do código estático em Scratch, para indicar o nível de pensamento computacional dos alunos.

Estou considerando uma amostra de 10 programas, sendo 5 deles escolhidos entre os que estão abaixo da média (Amostras Baixas) nas métricas LOC e McCabe e mais 5 escolhidos entre os que estão acima da média (Amostras Altas).

A Média Geral das amostras é de 17 pontos (MG=17,0). Considerando os dois grupos de amostras (5 Amostras Baixas e 5 Amostras Altas), as médias ficaram próximas da Média Geral, sendo a Média das Amostras Baixas (MAB=16,8) e Médias das Amostras Altas (MAA=17,2) ambas distantes 1,2% da média geral. Sendo assim, não faz sentido separar as amostras em dois grupos. As amostras e as médias podem ser observadas na Figura 24.

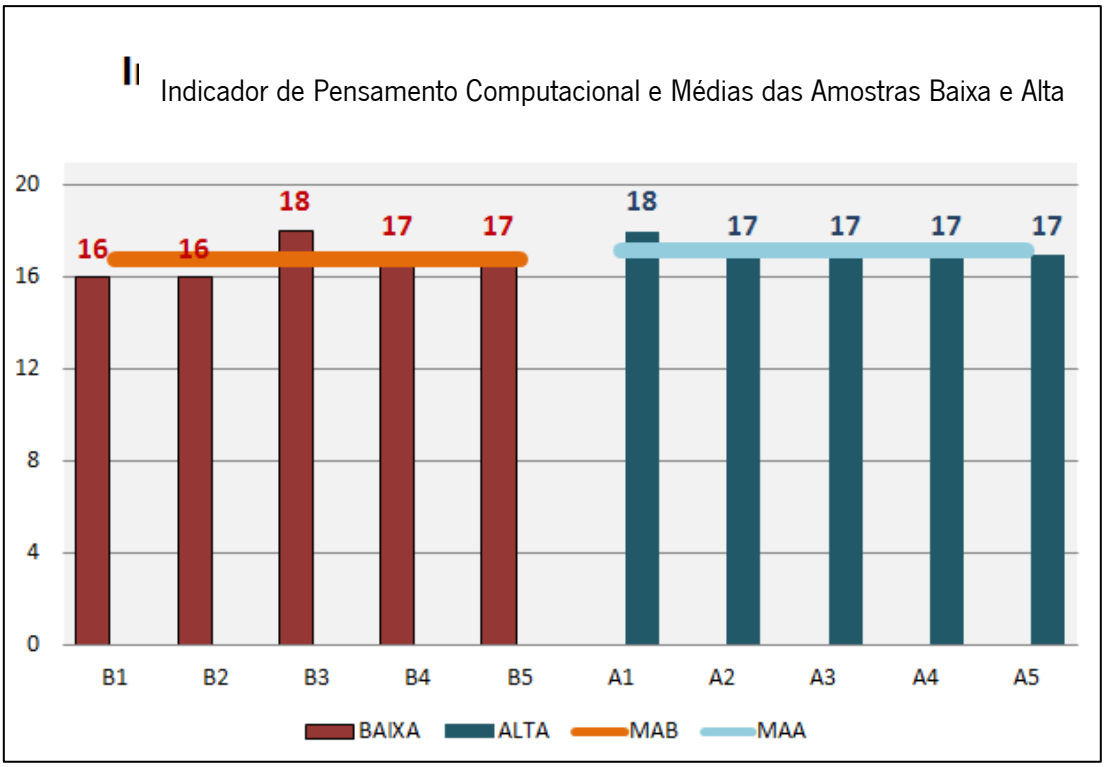


Figura 24 - Indicador de Pensamento Computacional e as Médias das Amostras Baixa e Alta

A seguir, são apresentados em gráficos do tipo radar os valores de cada uma das 10 amostras para os parâmetros que definem o Indicador do pensamento computacional. Cada parâmetro assume um valor de 0 a 3.

5.2.1.1 Lógica e Paralelismo

O parâmetro "Lógica" é avaliado como básico quando existe o uso do condicional SE, como em desenvolvimento quando existe o uso do condicional SE-SENÃO e como proficiente quando existem Operações lógicas (E, OU, NÃO). O parâmetro "Paralelismo" é avaliado como básico quando existem dois *scripts* em bandeira verde, como em desenvolvimento quando existem dois scripts quando uma tecla for pressionada, ou dois *scripts* para quando o objeto for clicado, e como proficiente quando existem dois ou mais *scripts* quando: recebe uma mensagem, ou cria clone, ou quando interface (cronômetro, áudio, vídeo) maior que um parâmetro, ou quando muda o pano de fundo. Os parâmetros de "Lógica" e "Paralelismo" assumem valores 3 para todas as amostras (ver Figura 25).

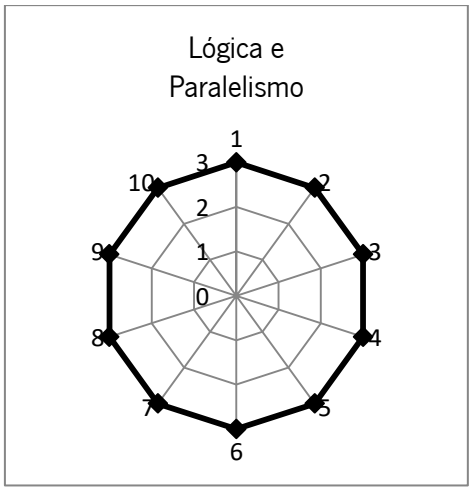


Figura 25 – Avaliação de Lógica e Paralelismo

5.2.1.2 Interatividade com o Usuário

O parâmetro "Interatividade com o Usuário" é avaliado como básico quando existe bloco iniciado com bandeira verde, como em desenvolvimento quando existe a verificação de tecla pressionada, ou objeto clicado, ou pedir resposta e esperar a entrada do teclado, ou verifica posição ou clique do mouse, e como proficiente quando existe a verificação, as interfaces de cronômetro, áudio, ou vídeo comparando como um parâmetro. O parâmetro "Interação com o Usuário" tem o valor 2 para todas as amostras. O valor 2 quer dizer que os projetos não utilizaram o indicador de volume de áudio como entrada de dados. O Dr.Scratch também pontua como 3 a entrada de vídeo, mas esta opção não está disponível na versão 1.4 na qual foram realizados os projetos analisados (ver Figura 26).

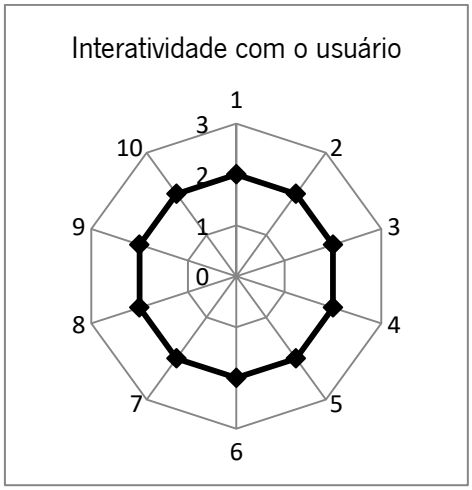


Figura 26 – Avaliação da Interatividade com o Usuário

5.2.1.3 Representação dos Dados

O parâmetro "Representação de Dados" é avaliado como básico quando existe a modificação nas propriedades dos objetos, como em desenvolvimento quando existe o uso de variáveis e como proficiente quando existe o uso de listas. No parâmetro "Representação de Dados" predomina o valor 2, com duas amostras com valor 3, significando que as listas de dados são raramente utilizadas (ver Figura 27).

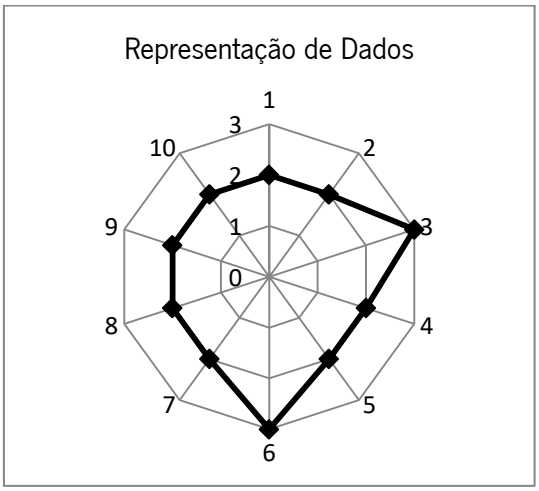


Figura 27 – Avaliação da Representação de Dados

5.2.1.4 Controle de Fluxo

O parâmetro "Controle de Fluxo" é avaliado como básico quando existe uma sequência de blocos, como em desenvolvimento quando existe o uso dos loops "Repita" ou "Sempre" e como proficiente quando existe o uso do loop "Repita até que". No parâmetro "Controle de Fluxo" predomina o valor 3 com apenas uma amostra com valor 2 (ver Figura 28).

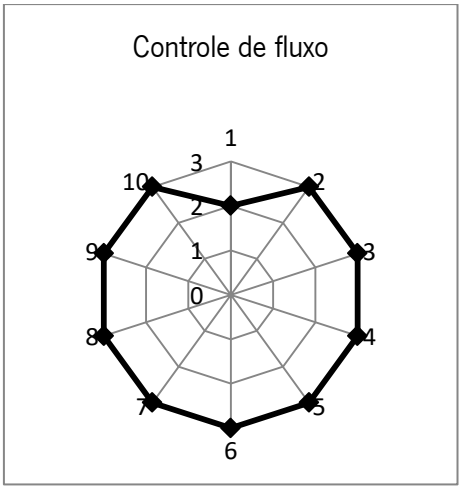


Figura 28 – Avaliação do Controle de Fluxo

5.2.1.5 Sincronização

O parâmetro “Sincronização” é avaliado como básico quando se usa o comando “espere”, como em desenvolvimento quando existe “envia” (sem espera) e recebe mensagem, ou “parar tudo”, ou “usar I comando pare”, e como proficiente quando se usa o comando “aguarda até que”, quando se muda pano de fundo, envia uma mensagem e espera. No parâmetro “Sincronização” predomina o valor 3 com apenas uma amostra com valor 2 (ver Figura 29).

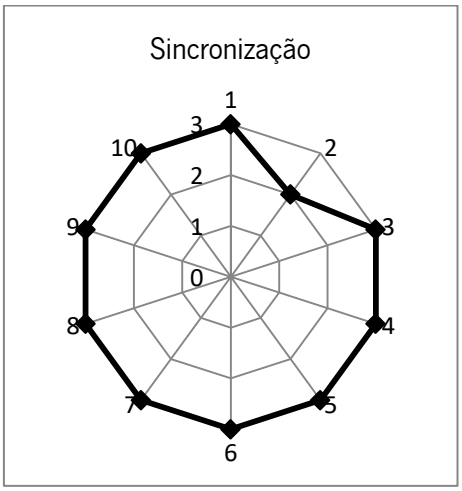


Figura 29 - Sincronização

5.2.1.6 Abstração

O parâmetro “Abstração” é avaliado como básico quando existem mais de um *script* e mais de um objeto, como em desenvolvimento quando existe Definição de Blocos e como proficiente quando existe a utilização de clones. O parâmetro de Abstração recebeu valor 1 em todas as amostras. O valor 1 indica que não foram utilizados os comandos de definição de blocos que caracterizam o valor 2, nem o uso de clones que caracterizam o valor 3. Porém, estes recursos não estão disponíveis na versão 1.4, logo, não seria possível que aparecessem nos projetos das amostras (ver Figura 30).

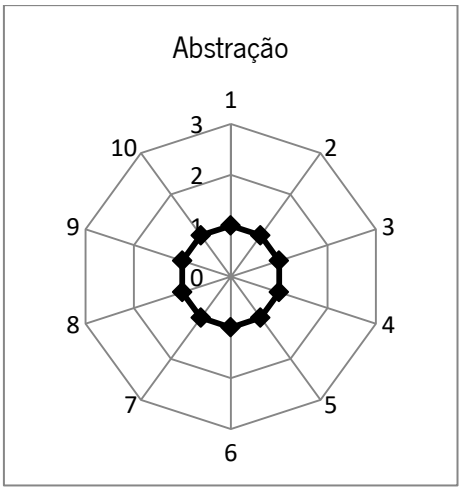


Figura 30 - Abstração

5.2.2 Incidências no processo de aprendizagem

A média de 17 pontos é alta, considerando o total de 21 pontos da metodologia Dr. Scratch e poderia ser considerada ainda mais alta uma vez que os valores 2 e 3 para o parâmetro Abstração não se aplicam aos projetos da versão 1.4. A entrada de teclado é a mais utilizada nos projetos, seguida da interface do mouse. O fato de o volume do áudio captado pelo microfone não ser utilizado não compromete a qualidade dos projetos apresentados. Por outro lado, a lista de dados é raramente utilizada e este fato me motiva a reforçar este item nos roteiros iniciais. Caso a lista continue sem utilização depois disso, isso pode acontecer devido a ser uma opção que os alunos fazem ao escolher sua forma de se expressar utilizando os programas em Scratch.

De uma forma geral, a média dos índices do pensamento computacional das amostras, utilizando Dr. Scratch, aponta para que os alunos de Introdução a Lógica de Programação chegam ao final com sucesso no aprendizado do pensamento computacional, usando a linguagem e o ambiente Scratch como forma de expressão.

5.2.3 Evidência documental do processo de aprendizagem proposto

O estudo de caso se limitou a turmas de Introdução à Lógica de Programação dos cursos de licenciatura em física e licenciatura em matemática, entre o ano de 2011, quando comecei a utilizar o Scratch, e o ano letivo de 2014. Este é o ambiente natural estudado, no qual utilizarei o estudo de caso para descrever e avaliar o conhecimento obtido na fase anterior inspirado na aito-biagrafia.

Na seção anterior fiz um relato histórico para chegar a um processo de aprendizagem utilizado para ensinar a disciplina de Introdução a Lógica de Programação. O curso foi dividido em três tipos de atividades: roteiros, prova e projetos. Em cada uma dessas fases os alunos realizaram atividades nas quais construíam artefatos e aprendiam com o processo, conforme preconiza a teoria construcionista (Papert & Harel, 1991). A dinâmica deste processo foi descrita na autobiografia. Nesta seção, avaliarei os processos de ensino através dos documentos que estão relacionados às tarefas dos roteiros, prova e projetos, para verificar se o objetivo de um aprendizado, para além da memorização de técnicas de programação, é possível de se atingir com este processo e se, ao mesmo tempo, existe a possibilidade de estimular a autonomia e a criatividade dos alunos. Tomarei a taxonomia de Bloom como padrão para avaliar a qualidade do processo de aprendizagem, baseado no estudo de Cynthia Selby (2014), crendo que o estímulo ao pensamento computacional será atingido caso sejam estimulados os processos cognitivos superiores da taxonomia de Bloom.

Minha intenção é analisar o processo de aprendizagem através dos documentos gerados pelos alunos e pelos apontamentos que eu recolhi durante o estudo. Para isso analiso as tarefas, isto é, os documentos que propõem as atividades desenvolvidas pelos alunos, relativas às atividades dos roteiros, prova e os projetos, usando a taxonomia de Bloom como referência.

5.2.3.1 Roteiros de atividades

As atividades estimuladas pelos roteiros acontecem no laboratório de informática, onde os alunos executam o procedimento descrito no roteiro. Depois de cada roteiro, um programa é construído pelo aluno, auxiliado pelos colegas, com suporte do professor. Por vezes o roteiro é iniciado em classe e finalizado em tempo extraclasse. Uma descrição inicial dos roteiros é dada na Tabela 26.

– *Objetivos didáticos dos Roteiros*

O programa Hello World é composto de apenas 2 comandos que são o chapéu “quando bandeira clicada” e o comando “diga [ola!] por 2 segundos”. Tem como objetivo mostrar que o programa é feito juntando comandos e orienta o aluno detalhadamente para salvar o programa depois de pronto.

No programa Gato Caminha**,** que faz o gato caminhar na tela da esquerda para a direita, é apresentado um bloco de comandos para serem copiados. Tem como objetivo realizar um

programa maior que Hello Word e fazer com que o aluno entenda que os comandos estão separados em grupos determinados pelas cores dos comandos. Também incentiva o aluno a fazer modificações, como mudança de cenários, e a interferir no código, explorando o ambiente de desenvolvimento Scratch com as seguintes provocações: "Faça o gato miar após a caminhada. Experimente outros fundos para o palco. Experimente colocar o gato em outras posições iniciais. Experimente acrescentar no programa o 

". Posteriormente, em aula expositiva, é explicado o loop *repita até* e a estrutura do contador que faz o gato andar 30 passos.

Tabela 26 - Descrição dos Roteiros

Nome do roteiro	Descrição
1) Hello Word	Faz o gato do Scratch dizer "Olá".
2) Gato Caminha	Faz o gato caminhar na tela da esquerda para a direita.
3) Andamento do Ciclista	Pede a distância e o tempo de um percurso e indica se a velocidade do ciclista foi Lenta, Moderada ou Alta.
4) Dacing Queen	Faz a personagem Cassy do Scratch dançar enquanto toca o som Hip Hop da biblioteca.
5) Voo do Morcego	Faz com que um morcego voe na parte superior da tela, seguindo a trajetória de uma função senoidal.
6) Carro	Faz um carro se movimentar na tela em uma paisagem em perspectiva, deslocando-se no eixo y e diminuindo de tamanho seguindo uma função linear.
7) Moeda	Simula um jogo no qual uma moeda se altera entre cara e coroa após o jogador apostar em uma das faces.
8) Dados	Semelhante ao programa Moeda. Existem dois dados e deve-se apostar na soma das faces.
9) Ôla	Posiciona uma série de clones em uma linha horizontal na tela e desloca um de cada vez formando uma onda como uma ôla (*hola*) de uma plateia de estádio de futebol.
10) Labirinto	Movimenta um objeto visto de cima usando as setas para deslocar para frente e para trás e virando no sentido horário e anti-horário.
11) Lançamento Obliquo	Solicita ângulo e velocidade inicial para lançamento de um objeto na tela que descreverá uma parábola da esquerda para direita, eventualmente atingindo o objeto alvo na parte direita da tela.

O programa Andamento do Ciclista pede a distância e o tempo de um percurso e indica se a velocidade do ciclista foi Lenta, Moderada ou Alta. Introduz o condicional *se*. Posteriormente é explicada a estrutura tradicional de programação que é a entrada de dados (distância e tempo), processamento (cálculo da velocidade) e saída (classificação da velocidade). Também é

relacionado com a classificação da velocidade e com a composição de intervalos reais, destacando os pontos limites.

Diferente dos roteiros anteriores, onde era preciso reproduzir o desenho do programa através do encaixe de comandos, os roteiros passam a ser escritos, de modo que o aluno tem que interpretar o roteiro para desenhar o programa. O comando *diga* pode aparecer como "use o comando diga para apresentar a mensagem Lento por 2 segundos" ou "comunique ao usuário que a velocidade é considerada Lenta". Ao final o aluno é incentivado a modificar o programa tentando diferentes tempos para *espera*, diferentes músicas e diferentes trajes.

O programa Dancing Queen faz a personagem Cassy do Scratch dançar enquanto toca o som hip hop da biblioteca do Scratch. Neste programa é passado o conceito de programação paralela, quando dois loops independentes são executados ao mesmo tempo, sendo um loop para a movimentação do personagem e outro para repetir a frase musical do hip hop. Também é introduzido o uso de sons como possibilidade para a programação.

O programa Voo do Morcego faz com que um morcego voe na parte superior da tela, seguindo a trajetória de uma função senoidal. Embora a função senoidal seja conteúdo do ensino médio, a maioria vê este conteúdo pela primeira vez nesta disciplina. É feita uma breve explicação sobre a função e como é adaptado o período, amplitude e nível médio para que a trajetória se dê na parte superior da tela. Contando com os conhecimentos do contador, expostos no programa Gato Caminha, utiliza-se um contador de 0 a 100 como variável independente e utiliza-se x e y como função desse contador. A equação do voo do morcego é a mesma para toda a classe. O roteiro solicita a implementação de um contador, sem dar detalhes da sua implementação.

O programa Carro faz um carro se movimentar na tela em uma paisagem em perspectiva, deslocando-se no eixo y e diminuindo de tamanho seguindo uma função linear. É semelhante ao Voo do Morcego, embora a função seja matematicamente mais simples (reta). Dependendo do desenho da paisagem os alunos têm pontos diferentes de partida e chegada, tendo que adaptar a função linear ao seu caso específico.

O programa Moeda simula um jogo no qual uma moeda se altera entre cara e coroa após o jogador apostar em uma das faces. É introduzido o comando *sorteie.* Nesse roteiro os valores 1 e 2 são associados a cara e coroa. São comparadas as variáveis *FaceEscolhida* e *FaceSorteada* para definir se o jogador ganhou a aposta e a face da moeda apresentada deve ser

coerente com a convenção adotada (1 para cara e 2 para coroa). Nesta atividade alguns erros são recorrentes, como não atualizar a face da moeda para o valor sorteado ou atualizar segundo a variável errada, ou se equivocar na hora de atualizar a variável *FaceEscolhida.* Sendo assim, os alunos precisam buscar coerência, fazendo o programa funcionar corretamente no processo de descrição-execução-reflexão-depuração. No final é solicitado que inclua o som de uma moeda girando, enquanto a moeda fica trocando de face.

O programa Dados é semelhante ao programa Moeda. Existem dois dados e deve-se apostar na soma das faces. Pergunto à classe quantas variáveis seriam necessárias. Costuma-se confundir a variável com o domínio, sendo assim, ao invés de uma variável por dado, costuma-se sugerir seis variáveis, como se fosse necessária uma variável para cada valor do domínio. Após a discussão, fechamos com uma variável para cada dado (FaceDado1, FaceDado2) e mais uma para a soma escolhida pelo usuário (SomaEscolhida). No roteiro há um desafio de que o segundo dado nunca deve apresentar o mesmo valor que o primeiro, implicando assim, em uma parte do algoritmo, que é necessário sortear o segundo dado em um loop repita até que o valor do segundo dado seja diferente do primeiro. O fragmento do programa que faz esta atividade é apresentado no roteiro, no entanto o aluno precisa analisar o código para conseguir encaixar no seu programa. Eu discuto esse requisito em sala de aula, de modo que eles já saiam com ideia de como funciona este fragmento de programa.

O programa Ôla posiciona uma série de clones de um objeto em uma linha horizontal na tela e desloca um de cada vez, formando uma onda como uma ôla (*hola*) de uma plateia de estádio de futebol. É necessária a criação de uma variável válida apenas para o objeto a ser clonado chamada de Identidade. Quando o comando ôla é recebido e um contador global bater com a identidade do objeto, o objeto faz o movimento desejado para ôla. Em seguida os alunos são estimulados a modificar o programa para posicionar os clones em linhas e colunas, em função do valor único armazenado em Identidade e fazer com que a ôla circule 4 vezes nos objetos das bordas (versão 1.4). Como não existe comandos de clonagem na versão do Scratch 1.4, a clonagem foi feita desenvolvendo um objeto matriz e duplicando manualmente o objeto matriz.

O programa Labirinto movimenta um objeto visto de cima usando as setas para deslocar para frente e para trás e virando no sentido horário e anti-horário. Com este roteiro, a

movimentação de objetos em função das setas do teclado e identificação de colisões de objetos são apresentadas, o que será muito útil para o desenvolvimento de jogos.

O programa Lançamento Oblíquo solicita o ângulo e a velocidade inicial para lançamento de um objeto na tela que descreverá uma parábola da esquerda para direita, eventualmente atingindo o objeto alvo na parte direita da tela. É uma aplicação da fórmula de lançamento oblíquo da física. A esta altura não há elementos novos significativos e os roteiros servem como reforço e treinamento do que já se sabe sobre programação com Scratch. Este roteiro também serve como um dos mecanismos de jogos ensinados no curso.

5.2.3.2 Provas de conhecimentos

As provas foram adaptadas à mídia de papel e lápis, tomando como princípio que cada mídia é uma nova mediação estimulando uma nova forma de pensar (Borba, 2002). As provas são programas apresentados aos alunos para serem explicados, relacionando o código (causa) com o funcionamento do programa (efeito).

Esta atividade é realizada em três etapas:

1. Treinamento em sala de aula.
2. Treinamento em casa e verificação por pares.
3. Aplicação da Prova.

Para o treinamento em sala de aula, eu mostro um programa e peço que a classe explique o seu funcionamento. À medida que as opiniões vão aparecendo, eu vou chamando a atenção para os detalhes e vou refazendo as frases oralmente, tentando melhorar a narrativa. Depois desse laboratório coletivo, peço para que eles escrevam a explicação do programa. Após a escrita individual, eu peço para que alguns voluntários leiam o que escreveram. A principal mensagem é que as relações entre causa e efeito estejam presentes no texto.

Na fase de treinamento em casa e verificação por pares, eu entrego dois programas para que os alunos escrevam a explicação em casa. Depois promovo em sala de aula uma troca das explicações e peço que corrijam o que o colega escreveu, destacando três tipos de frases.

- As que relacionam causa e efeito;
- As que têm erros conceituais;
- As que estão confusas ou sem sentido.

Durante esta aula os próprios alunos despertam para dúvidas conceituais tais como se o comando *repita até* é considerado loop ou condicional. Na aula seguinte eles procuram os autores dos textos explicativos e dão o retorno da análise que fizeram.

Finalmente, é realizada a aplicação da prova com dois pequenos programas para serem explicados em duas horas.

5.2.3.3 Projetos de programação em Scratch

São realizados dois projetos com uma complexidade crescente. Primeiro é realizado um pequeno projeto de uma semana antes do grande projeto de três semanas.

Como projeto de uma semana, proponho uma animação com o Scratch. O tema é definido pela classe através de sugestões e votação. No projeto de animação não existe interação com o usuário e o fluxo do programa é único, isto é, as cenas acontecem sempre na mesma sequência. Para o projeto de três semanas o tema é livre. A intenção é que se faça um jogo, com uma narrativa, usando mecanismos clássicos de jogos eletrônicos. Os mecanismos são ensinados antes do projeto do jogo e a narrativa não precisa ser sequencial, podendo uma cena se repetir ou não aparecer, dependendo do fluxo tomado a partir dos resultados no jogo e das interações com o usuário. A interação com o usuário, a narrativa com fluxos alternativos e a liberdade de escolha do tema colocam o aluno frente a desafios maiores que a animação. A entrega ou apresentação dos projetos são momentos de socialização quando os alunos mostram e explicam seus projetos uns aos outros.

Para o planejamento das atividades é desenvolvido um *storyboard*, representando a animação ou o jogo. O *storyboard* fornece uma representação do trabalho a ser realizado, possibilitando a discussão, a antecipação de dificuldades e as alterações no projeto antes de iniciar a programação.

Os projetos são feitos em duplas. A interação, a solidariedade e a busca por ajuda dentro e fora do grupo são incentivadas para que o projeto tenha um mínimo de complexidade. Estabeleci algumas regras através de pré-requisitos e parâmetros para a avaliação que descrevo a seguir.

Pré-requisitos:

a. O projeto deve utilizar a estrutura de chamar uma cena de cada vez no palco com o comando anuncia e espere.

b. Será necessário que o projeto tenha um mínimo de 5 cenas.
c. Será exigido que o projeto tenha um mínimo de 2 mecanismos de jogos implementados.
d. O projeto deve apresentar uma narrativa.
e. Um *storyboard* deve ser produzido no início do projeto e servirá de base para o acompanhamento do projeto.

Parâmetros de Avaliação:

a. *Storyboard*;
b. Evolução individual nas reuniões de acompanhamento;
c. Complexidade do Projeto;
d. Apresentação do projeto.

No projeto de três semanas, eu peço para que os alunos estabeleçam metas semanais para os seus próprios projetos. Estas metas e o acompanhamento individual de cada grupo são feitos uma vez por semana de acordo com a seguinte orientação.

- Finalizar o *storyboard* na primeira semana;
- Na segunda semana cerca de 50% ou mais das cenas devem estar prontas;
- Na terceira semana deve-se rever o projeto para conciliar o que se pretende realizar com o prazo que resta.

5.3 Síntese da avaliação do processo de aprendizagem

Para avaliar o processo evidenciado pelos documentos de roteiros, provas e projetos, utilizei a taxonomia de Bloom, levando em consideração as dimensões Processos Cognitivos e Tipos de Conhecimento. Atribuí a cada um dos Processos Cognitivos um valor (1 Lembrar, 2 Entender, 3 Aplicar, 4 Analisar, 5 Avaliar, 6 Criar) e a cada um dos Tipos de Conhecimento uma letra (A Factual, B Conceitual, C Processual, D Metacognitivo). Como consequência, o cruzamento das células identificadas por uma letra e um número fornece uma expressão de avaliação, por exemplo, B2 - Entendimento Conceitual. Uma matriz com as células representando as dimensões dos processos cognitivos e dos tipos de conhecimento caracterizam o instrumento de avaliação correspondente à aplicação da taxonomia de Bloom.

5.3.1 Mapeando os roteiros à taxonomia de Bloom

Cada roteiro pode contribuir para uma ou mais células da taxonomia. A fim de identificar as células que são estimuladas por cada roteiro, tomarei como base os seus objetivos didáticos (ver item 5.2.3.1). Para identificar as células que são estimuladas pelos roteiros seguirei os seguintes passos:

1. Rever os objetivos didáticos para identificar as subclasses dos processos cognitivos e tipos de conhecimento estimulados (Coluna *Adequação às subclasses* da Tabela 27 até a Tabela 37).
2. Com base nas subclasses, mapear as células estimuladas pelos roteiros (ver coluna *Célula* da Tabela 27 até a Tabela 37).
3. Identificar as ocorrências dos estímulos nos roteiros (ver Tabela 38).

O processo de aprender a programar computadores com auxílio de roteiros é naturalmente referenciado como um processo cognitivo *Aplicar* e com o tipo de conhecimento Processual. Por isso, aparece na Adaptação das Descrições dos Roteiros à taxonomia de Bloom recorrentemente a frase "Executar o roteiro (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos)".

O roteiro indica como deve ser feito o programa, mas o aluno deve interpretá-lo. Entre a interpretação e a produção do programa final, o programador passa pelo processo natural de descrição-execução-reflexão-depuração (Maltempi & Valente, 2000). Este processo refletiu-se na adaptação, estimulando as subclasses de conhecimento Avaliar e Processual, quando solicito "Verificar o funcionamento do programa de acordo com o projetado (Técnicas, Modelos e Específicos)."

Tabela 27 - Mapeamento dos Roteiros - Hello World

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Hello Word	Inferir que um programa é composto por uma união de comandos. (Detalhe e Elemento Específico)	2 Entender	A Factual	A2
Hello Word	Executar o processo de salvar um programa. (Habilidade e Algoritmo Específico)	2 Aplicar	A Factual	A3
Hello Word	Executar o roteiro. (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos)	3 Aplicar	B Conceitual	B3

Tabela 28 – Mapeamento dos Roteiros –Gato Caminha

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Gato Caminha	Reconhecer o comando loop. (Terminologia e Detalhe e Elemento Específico)	1 Lembrar	A Factual	A1
Gato Caminha	Reconhecer a estrutura de um contador. (Técnicas e Métodos Específicos)	1 Lembrar	C Processual	C1
Gato Caminha	Inferir que os comandos estão separados por classes. (Classificação e Categorização)	2 Entender	B Conceitual	B2
Gato Caminha	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	C Processual	C2
Gato Caminha	Executar o roteiro. (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos).	3 Aplicar	B Conceitual	B3
Gato Caminha	Produzir modificações nos programas dados. (Princípios e generalizações com relação a interface e conhecimento sobre a própria cognição sobre a ferramenta)	6 Criar	B Conceitual	B6
Gato Caminha	Produzir modificações nos programas dados. (Princípios e generalizações com relação a interface e conhecimento sobre a própria cognição sobre a ferramenta)	6 Criar	D Metacognitivo	D6

Tabela 29 - Mapeamento dos Roteiros – Andamento do Ciclista

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Andamento do Ciclista	Reconhecer comando de entrada. (Factual)	1 Lembrar	A Factual	A1
Andamento do Ciclista	Reconhecer a estrutura de entrada, processo e saída de um programa. (Terminologia e Detalhe de Elemento Específico)	1 Lembrar	A Factual	
Andamento do Ciclista	Entender as relações entre a sequência de comandos se e os intervalos reais. (Teoria, Modelos e Estruturas)	2 Entender	B Conceitual	B2
Andamento do Ciclista	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)			
Andamento do Ciclista	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	C Processual	C2

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Andamento do Ciclista	Executar o roteiro. (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos)	3 Aplicar	B Conceitual	B3
Andamento do Ciclista	Verificar o funcionamento do programa de acordo com o projetado. (Técnicas, Modelos e Específicos)	5 Avaliar	C Processual	C5

Tabela 30 - Mapeamento dos Roteiros – Dacing Queen

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Dacing Queen	Reconhecer a programação em paralelo. (Teoria Modelos e Estruturas; Técnicas e Métodos Específicos)	1 Lembrar	B Conceitual	B1
Dacing Queen	Reconhecer a programação em paralelo. (Teoria Modelos e Estruturas; Técnicas e Métodos Específicos)	1 Lembrar	C Processual	C1
Dacing Queen	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	B Conceitual	B2
Dacing Queen	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	C Processual	C2
Dacing Queen	Executar o roteiro. (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos)	3 Aplicar	B Conceitual	B3
Dacing Queen	Produzir modificações nos programas dados. (Princípios e generalizações com relação a interface e conhecimento sobre a própria cognição sobre a ferramenta)	6 Criar	B Conceitual	B6

Tabela 31 - Mapeamento dos Roteiros – Voo do Morcego

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Voo do Morcego	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	B Conceitual	B2
Voo do Morcego	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	C Processual	C2
Voo do Morcego	Traduzir (interpretar) as fórmulas matemáticas em comandos Scratch.	2 Entender	C Processual	C2

	(Técnica e Métodos Específicos)			
Voo do Morcego	Executar o roteiro (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos).	3 Aplicar	B Conceitual	B3
Voo do Morcego	Verificar o funcionamento do programa de acordo com o projetado. (Técnicas, Modelos e Específicos)	5 Avaliar	C Processual	C5

Tabela 32 - Mapeamento dos Roteiros - Carro

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Carro	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	B Conceitual	B2
Carro	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	C Processual	C2
Carro	Traduzir (interpretar) as fórmulas matemáticas em comandos Scratch. (Técnica e Métodos Específicos)			
Carro	Executar o roteiro. (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos)	3 Aplicar	B Conceitual	B3
Carro	Implementar uma função linear para o deslocamento do carro. (Técnica e Métodos Específicos)	3 Aplicar	C Processual	C3
Carro	Verificar o funcionamento do programa de acordo com o projetado. (Técnicas, Modelos e Específicos)	5 Avaliar	C Processual	C5

Tabela 33 - Mapeamento dos Roteiros - Moeda

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Moeda	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	B Conceitual	B2
Moeda	Entender o projeto (modelo, design) do programa proposto. (Teoria, Modelos e Técnicas)			
Moeda	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	C Processual	C2

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Moeda	Executar o roteiro. (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos).	3 Aplicar	B Conceitual	B3
Moeda	Verificar o funcionamento do programa de acordo com o projetado. (Técnicas, Modelos e Específicos)	5 Avaliar	C Processual	C5
Moeda	Produzir modificações nos programas dados. (Princípios e generalizações com relação a interface e conhecimento sobre a própria cognição sobre a ferramenta)	6 Criar	B Conceitual	B6

Tabela 34 - Mapeamento dos Roteiros - Dados

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Dados	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	B Conceitual	B2
Dados	Entender o projeto (modelo, design) do programa proposto. (Teoria, Modelos e Técnicas)			
Dados	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	C Processual	C2
Dados	Executar o roteiro. (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos)	3 Aplicar	B Conceitual	B3
Dados	Implementar um novo algoritmo semelhando ao da moeda em outro programa. (Técnica e Métodos Específicos)	3 Aplicar	C Processual	C3
Dados	Verificar o funcionamento do programa de acordo com o projetado. (Técnicas, Modelos e Específicos)	5 Avaliar	C Processual	C5
Dados	Produzir modificações nos programas dados. (Princípios e generalizações com relação a interface e conhecimento sobre a própria cognição sobre a ferramenta)	6 Criar	B Conceitual	B6

Tabela 35 - Mapeamento dos Roteiros - Ôla

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Ôla	Reconhecer uma variável em um objeto alocada apenas para este objeto. (Técnica e Métodos Específicos)	1 Lembrar	C Processual	C1
Ôla	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	B Conceitual	B2
Ôla	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	C Processual	C2
Ôla	Traduzir (interpretar) as fórmulas matemáticas em comandos Scratch. (Técnica e Métodos Específicos)			
Ôla	Executar o roteiro. (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos).	3 Aplicar	B Conceitual	B3
Ôla	Verificar o funcionamento do programa de acordo com o projetado. (Técnicas, Modelos e Específicos)	5 Avaliar	C Processual	C5
Ôla	Produzir modificações nos programas dados. (Princípios e generalizações com relação a interface e conhecimento sobre a própria cognição sobre a ferramenta)	6 Criar	B Conceitual	B6

Tabela 36 - Mapeamento dos Roteiros - Labirinto

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Labirinto	Reconhecer o uso das setas para deslocar objetos. (Técnica e Métodos Específicos)	1 Lembrar	C Processual	C1
Labirinto	Reconhecer colisão de objetos. (Técnica e Métodos Específicos)			
Labirinto	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	B Conceitual	B2
Labirinto	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	C Processual	C2
Labirinto	Executar o roteiro. (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos)	3 Aplicar	B Conceitual	B3

Labirinto	Verificar o funcionamento do programa de acordo com o projetado. (Técnicas, Modelos e Específicos)	5 Avaliar	C Processual	C5

Tabela 37 - Mapeamento dos Roteiros – Lançamento Obliquo

Roteiro	Adequação as subclasses	Processo Cognitivo	Tipo de Conhecimento	Célula
Lançamento Oblíquo	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	B Conceitual	B2
Lançamento Oblíquo	Entender o projeto (modelo, design) do programa proposto.	2 Entender	B Conceitual	
Lançamento Oblíquo	Interpretar a escrita dos roteiros e codificar o programa desejado. (Princípios e Generalizações; Habilidade e Algoritmos Específicos)	2 Entender	C Processual	C2
Lançamento Oblíquo	Traduzir (interpretar) as fórmulas matemáticas em comandos Scratch. (Técnica e Métodos Específicos)	2 Entender	C Processual	C2
Lançamento Oblíquo	Executar o roteiro. (Habilidade e Algoritmos Específicos; Técnicas e Métodos Específicos).	3 Aplicar	B Conceitual	B3
Lançamento Oblíquo	Verificar o funcionamento do programa de acordo com o projetado. (Técnicas, Modelos e Específicos)	5 Avaliar	C Processual	C5

Tabela 38 - Ocorrência dos Processos Cognitivos e Tipos de Conhecimento nos Roteiros

Tipos de Conhecimento	Processos Cognitivos					
	1 Lembrar	2 Entender	3 Aplicar	4 Analisar	5 Avaliar	6 Criar
A Factual	2	1	1			
B Conceitual	1	10	11			5
C Processual	4	11	2		8	
D Metacognição						1

5.3.2 Implicações de Bloom: uma visão focada

Pela análise da Tabela 38, nota-se que a região delimitada pelas células A1:C3, é a mais estimulada pelos roteiros. A célula C5 aparece estimulada devido ao ciclo natural da programação de computadores descrição-execução-reflexão-depuração.

Estimular as células A1:C3 está de acordo com o planejamento do curso, no qual a fase dos roteiros é de caráter propedêutico. O aluno deve ter habilidade de programar antes de ser criativo utilizando a programação. Em analogia com a música, um músico não se faz músico sem a habilidade com seu instrumento. Só depois de aprender música e dominar seu instrumento é que ele será capaz de ser criativo na música (Morais, 2012). A programação tem este caráter prático em que se faz necessário dominar o ambiente de desenvolvimento, os comandos e, por fim, os conceitos e algoritmos da programação de computadores. Para o programador isto seria dominar o seu instrumento, para depois poder se expressar com criatividade. Embora aconteça de forma secundária, a experiência mostra que as pequenas alterações solicitadas pelos roteiros estimulam a criatividade e o ousar ser criativo, evitando a apatia, no futuro, quando os alunos se depararem com desafios maiores com relação a criatividade. Nesse pressuposto, podemos considerar que as células B6 e D6 são fracamente estimuladas.

Um aspecto que a taxonomia não mostra é a complexidade das tarefas. Por exemplo, produzir modificações no programa estimula a célula B6 (Criar/Conceitual), mas alterar o cenário no roteiro Gato Caminha eu considero uma modificação de baixa complexidade. Por oposição, alterar o programa Dados para que o segundo dado não apareça com a mesma face do primeiro, eu considero uma alteração de média complexidade. Os desafios de alta complexidade aparecerão nos projetos em uma fase posterior no curso da disciplina. Observando a descrição dos roteiros, nota-se um crescente na complexidade proposta.

Outro aspecto que não aparece nessa análise é o relativo aos aprendizados com mínima intervenção (Mitra, 2000). Ao usar as ferramentas, aprende-se, por exemplo, que os comandos que fazem um objeto se movimentar estão na aba Movimentos e estão todos em azul, ou que para separar as sequências de comandos, todos os comandos abaixo do selecionado são arrastados juntos. Estes conhecimentos são adquiridos pela interação com o roteiro, o computador, os seus pares e o professor. Neste processo todas as pessoas envolvidas geram conhecimento e não só o professor, haja visto que aprendi com eles que é possível fazer o programa rodar lentamente num processo passo a passo, ou que a importação de imagens GIF animadas gera uma série de imagens no objeto que importou o GIF.

Na explicação dos programas, vários processos interagem para produzir o texto explicativo a partir de um programa, como a análise do programa diferenciando partes do código pela sua

função, reconhecendo técnicas e métodos utilizados, elaboração do texto explicativo. O mapeamento desses processos segundo a taxonomia de Bloom é representado na Tabela 39.

Tabela 39 - Mapeamento dos Processos Existentes na Explicação do Programa

Processo	Classificação	Célula
Análise do programa diferenciando partes do código pela sua função.	Diferenciar Teorias Modelos e Estruturas. (Análise/Conceitual)	B4
	Diferenciar Técnicas e Métodos Específicos. (Análise/Processual)	C4
Reconhecendo técnicas e métodos utilizados.	Reconhecer Teorias Modelos e Estruturas. (Lembrar/Factual)	A1
	Reconhecer Técnicas e Métodos Específicos. (Lembrar/Processual)	B1
Elaboração do texto explicativo.	Produzir Princípios e Generalizações. (Criar/Conceitual)	B6

Os projetos, de forma geral, se adequam ao processo cognitivo Criar, pelo seu caráter de originalidade, usando os conhecimentos de programação. A elaboração do *storyboard* se encaixa na subclasse planejando. O artefato *storyboard* pode ser considerado um documento conceitual de generalizações e um modelo de estruturas. Porém, ao discutir com seus pares, os alunos despertam para as estratégias adotadas no desenvolvimento, considerando os pontos fortes e fracos dos parceiros, tomando consciência da cognição de si mesmos e do parceiro de trabalho. Por isso, além do conhecimento conceitual (célula B6), o *storyboard* também contribui para a metacognição (D6)

O acompanhamento de projeto é essencialmente o confronto do que foi realizado com o que era esperado, tomando o *storyboard* como base, rediscutindo as estratégias e modificando o planejado, caso se avalie como necessário. Deste modo, podemos classificar esta atividade como pertencente à célula D5 Avaliar/Metacognição.

A programação do projeto é equivalente a produzir um algoritmo, ou seja, pode ser classificada como sendo indício da subclasse produzindo, com relação à cognição, e um algoritmo estaria na classe procedural da dimensão do conhecimento. O uso do ciclo descrição-execução-reflexão-depuração e as dificuldades encontradas na programação e suas respectivas soluções caracterizam as subclasses verificando e gerando. Assim a programação do projeto se encontra nas células C5 e C6.

5.3.3 Implicações de Bloom: uma visão geral

Para resumir o mapeamento das atividades do curso em relação à taxonomia de Bloom, observemos a Tabela 40.

Tabela 40 - Mapeamento das Atividades na Taxonomia de Bloom

Tipos de Conhecimento	Processos Cognitivos					
	1 Lembrar	2 Entender	3 Aplicar	4 Analisar	5 Avaliar	6 Criar
A Factual	Roteiro Prova	Roteiro	Roteiro			
B Conceitual	Roteiro Prova	Roteiro	Roteiro	Prova		Roteiro Prova Projeto
C Processual	Roteiro	Roteiro	Roteiro	Prova	Roteiro Projeto	Projeto
D Metacognição					Projeto	Roteiro Projeto

O curso de Introdução à Lógica de Programação é formado por três etapas: os roteiros, a prova e os projetos. Os roteiros são responsáveis pela parte introdutória para entender os conceitos e adquirir competências de utilização do ambiente de desenvolvimento, bem como na escrita e depuração de programas. As células A1, A2, A3, B1, B2, B3, C1, C2, C3 e a célula C5 são fortemente estimuladas, enquanto as células B6 e D6 são fracamente estimuladas.

As atividades relacionadas com a prova destacam o entendimento dos programas escritos, estimulando a análise, tendo que relembrar conceitos de programação para produzir um texto explicativo com vocabulário adequado. As atividades relacionadas com a prova estimulam as células A1, B1, B4, C4 e B6.

Os projetos são as atividades finais do curso. Neles os alunos são impelidos a criar programas originais, são levados a planejar e a executar uma proposta que sejam capazes de realizar. Neste processo, os alunos precisam de desenvolver um conhecimento da sua capacidade e da capacidade do grupo. Também são levados a refletir sobre o próprio trabalho e sobre os prazos, nas aulas de acompanhamento do projeto. As atividades de projeto estimulam as células C5, D5, B6, C6 e D6.

A análise do processo de aprendizagem adotado para o ensino da disciplina de Introdução a Lógica de Programação mostrou que as atividades associadas aos roteiros, à prova e aos projetos, estimulam todos os processos cognitivos e tipos de conhecimento

definidos na taxonomia de Bloom. Estes estímulos acontecem de forma gradual com os roteiros atuando nas células mais baixas (A1 : C3), a prova atuando nas células intermediárias (B4, C4) e os projetos atuando nas células mais altas (C5 : D6).

5.4 Em jeito de balanço da investigação

No processo de ensino de programação deve-se objetivar que o aluno seja criativo usando a programação como forma de expressão. A necessidade da criatividade traz como pressupostos a autonomia e, consequentemente, a liberdade de experimentação (Morais, 2012). A estratégia do conteúdo básico ensinado por roteiros em uma perspectiva sócio-interacionista, ou seja, os alunos auxiliados pelos colegas, além do professor, é um exemplo de prática bem sucedida. Os roteiros e o ambiente colaborativo promovem zonas de desenvolvimento proximal diferentes, de maneira que um aluno que ajuda em um ponto de um roteiro pode ser ajudado em outro.

A retirada dos erros de sintaxe do processo de escrita do código é a maior vantagem do uso do Scratch como ambiente de desenvolvimento. Muito tempo foi ganho e muita frustação evitada com a eliminação dos erros de sintaxe. O Scratch resulta em aplicações gráficas que, assim como o a linguagem Logo, fornece realimentação (*feedback*) para o aluno-programador saber se o programa está funcionando a contento, criando um ambiente de experimentação que Seymour Papert nominou de micromundo (Jonassen, 2007).

No projeto final exige-se um nível de criatividade que os alunos não estão acostumados a usar, como observei nas primeiras turmas, no desenvolver do projeto final em Scratch. Porém, a criatividade pode ser incentivada. Nesse pressuposto solicitei modificações nos programas gerados com os roteiros. Depois, na animação, eles tinham o apoio de um tema comum e, no projeto final, eles desenvolviam um programa cujo tema, planejamento e execução são definidos totalmente pela dupla executora.

Penso que a dupla é a melhor equipe que se possa ter para realizar um projeto, porque assim os alunos têm com quem discutir as suas ideias e aprendem a trabalhar colaborativamente. Eles se cobram e se motivam na realização do trabalho. Trabalhar em duplas reduz a possibilidade de existir uma pessoa do grupo alheia ao processo, sendo a melhor aproximação possível para uma avaliação individualizada.

Em termos afetivos, as modificações dos roteiros personalizam os trabalhos, fazem com que os alunos o reconheçam como sendo seu, promovendo uma identificação. A necessidade de

perguntar aos colegas nos primeiros roteiros cria laços afetivos duradouros. O projeto de animação com o tema escolhido pela classe e os vídeos com os trabalhos e fotos dos colegas, também reforçam os laços afetivos. Por tudo isso, o outro está sempre presente na aprendizagem da programação. A visão do outro é um dos requisitos a criatividade (Morais, 2012), além de possibilitar o diálogo proposto pelo D3NA – Diversão, Diálogo, Desafio, Narrativa e Aventura (Meira & Pinheiro, 2012). Também constatamos a *diversão* ao realizar e mostrar aos colegas seus trabalhos. A produção das *narrativas* na animação e no projeto final faz parte da *aventura* de participar da disciplina. Apesar de divertido, os alunos não consideram Introdução a Lógica de Programação uma disciplina fácil, mas sim uma que exige muito raciocínio. Sendo assim, talvez possamos dizer que os alunos estão trabalhando naquela zona de bastante esforço com recompensa e aprendizado adequados que Papert chamou de *Hard-Fun*.

Para se atingir o pensamento complexo é necessário conceber ideias, tomar decisões e resolver problemas. Observei estas atitudes durante as aulas, um pouco nos roteiros e bastante nos projetos, pelo que, penso que a proposta do curso culminando com os projetos leva os alunos a exercitar o pensamento complexo.

Estas conclusões vindas da autobiografia apresentadas aqui se baseam na observação sistemática evidenciadas. Apesar do processo de aprendizagem proposto ter sido criado a partir de uma iniciativa individual e subjetiva com um método de observação, planejamento, intervenção e avaliação, ele pode ser avaliado de forma mais sistemática com relação ao pensamento complexo e seu valor cognitivo. A ferramenta Dr Scratch apontou níveis altos do pensamento computacional para todos os programas da amostra de projetos, apesar da ferramenta não se mostrar adequada à avaliação do parâmetro de abstração dos projetos realizados com a versão 1.4 do Scratch. A análise dos documentos que propõem as atividades de Roteiros, Provas e Projetos mostraram ganhos cognitivos em diferentes regiões da análise bidimensional da taxonomia de Bloom. As três atividades funcionam de maneira complementar e gradual, de modo a incentivar nos alunos competências cognitivas de níveis mais baixos na atividade dos Roteiros, competências cognitivas intermediárias na atividade da Prova e competências cognitivas elevadas nas atividades de Projeto. Conclui-se que o processo de aprendizagem proposto estimula o pensamento computacional como evidenciam as métricas apresentadas pela ferramenta Dr. Scratch sobre a amostra dos programas desenvolvidos pelos

alunos. As atividades propostas também estimulam níveis altos de cognição, como evidenciou a análise dos documentos segundo a taxonomia de Bloom.

6 Considerações finais

Neste capítulo revisitarei a investigação, revendo conceitos como o do pensamento computacional e de aprendizagem. Relembro as descobertas da investigação e como o processo estabelecido influenciou o comportamento dos alunos nos aspectos de autonomia e criatividade. Destaco os pontos para reprodução e adaptação do processo de aprendizagem estabelecido e aponto para futuras pesquisas. Por fim, me proponho a responder a pergunta de pesquisa a partir dos conhecimentos obtidos com a investigação.

6.1 Retrospectiva da investigação realizada

Neste trabalho me dispus a investigar o processo de aprendizagem de modo a contribuir para uma melhor formação dos meus alunos, que serão futuros professores de matemática, partindo do seguinte problema de investigação:

Como conduzir um processo de aprendizagem em aulas regulares de programação de computadores, de modo a contribuir para o desenvolvimento do pensamento computacional na formação de futuros professores de matemática?

6.1.1 O Pensamento Computacional como meta de aprendizagem

O pensamento computacional serve como meta de aprendizagem que reúne capacidades e valores que considero importantes para formação dos alunos, sendo esta opinião validada por instituições como o Ministério da Educação no Brasil (Ministério da Educação, 2000, 2002), o National Research Council nos Estados Unidos (National Research Council, 2010, 2011) e a The Royal Society no Reino Unido (Furber, 2012).

Segundo Jeannette Wing, o pensamento computacional "são os processos de pensamento que envolvem a formulação de um problema expressando suas soluções, de tal forma que um computador ou humano possam executar de maneira efetiva (2014, p. 1). Embora muitos cientistas concordem em vários pontos, as várias definições de pensamento computacional divergem de modo a enfatizar quatro maneiras de ver este conceito: (1) Forma de Ver e

Conhecer; (2) Maneiras ou Capacidades de Realizar; (3) Método de Investigação; (4) Forma de Colaboração (National Research Council, 2011). (1) Forma de Ver e Conhecer: A metáfora do Scratch de palco ao fundo e de atores interagindo ajudam a formar a ideia de que a complexidade do mundo pode ser decomposta em partes menores e menos complexas, para efeito de manipulação do problema maior. Ao realizar as animações e os jogos as cenas são documentadas nos Storyboard e, em cada quadro do Storyboard, o aluno identifica os objetos (ou atores) e suas interações desejadas. Deste modo o projeto e sua narrativa são quebrados em partes menores e menos complexas. (2) Maneiras ou Capacidades de Realizar: Esta vertente valoriza a capacidade humana de realizar, como fizeram os alunos ao levar a cabo seus projetos segundo o que foi planejado, no prazo estabelecido. (3) Método de Investigação: Esta vertente valoriza as pesquisas realizadas estudando objeto através dos modelos implementados nos computadores. Embora os alunos consigam realizar suas simulações, elas não tem a intensão de estudar um determinado objeto. As simulações são utilizadas apenas para contar uma história. (4) Forma de Colaboração: Esta vertente, que valoriza a comunicação e a sincronização das atividades realizadas, também foi contemplada nos projetos na medida em que o trabalho em equipe demanda a distribuição das tarefas para realizar artefatos e a junção desses artefatos no projeto finalizado. Cada dupla estabelece suas estratégias. Alguns usam fazer um perfil comum no site do Scratch e acrescentam seus artefatos em um projeto *online*; outros usam o editor *offline* e sincronizam na classe ou em outra reunião presencial; outros elegem um integrador e o outro parceiro envia os artefatos para este primeiro integrar ao projeto.

Existem quatro principais argumentos para incentivar o pensamento computacional: (1) O Pensamento Computacional ajuda a ser criativo; (2) O Pensamento Computacional ajuda a resolver problemas em colaboração com outros; (3) O professor de matemática precisa de uma formação em tecnologia; (4) O Pensamento Computacional viabiliza múltiplas carreiras profissionais (Tucker et al., 2003). Os dois primeiros argumentos (criatividade e resolução de problemas em colaboração) são plausíveis pelo que notei durante a investigação, uma vez que, no desenvolvimento dos projetos, os alunos se utilizam da criatividade e da resolução de problemas. Usam a criatividade para fazer a história a ser contada e também para resolver os problemas. Os problemas são os desafios que eles mesmos se colocam ao criar a história e também para as retiradas dos erros de programação (*debug*), além dos metaproblemas como definição das tarefas a serem realizadas, a divisão do trabalho e prazos.

O pensamento computacional já é ensinado nas escolas nas áreas de ciência, tecnologia, engenharia e matemática, denominadas pela sigla em inglês STEM (Science, Technology, Engineering, and Mathematics) e nas disciplinas relacionadas às STEM são valorizadas as competências com Dados e Informações, Simulação, Resolução de Problemas e Pensamento Sistêmico (Weintrop et al., 2016).

Nos Parâmetros Curriculares Nacionais do Brasil (PCN) se reconhecem capacidades comuns entre o ensino de matemática e o pensamento computacional que podem ser organizadas em três grupos: (1) articulação de símbolos; (2) Identificação de padrões e regularidades; (3) Construção de modelos representativos e explicativos (Barcelos & Silveira, 2012). Ao programar os alunos articularam os símbolos da linguagem com o propósito de realizar os projetos. Os mecanismos de jogos e as técnicas de animação são padrões identificados e reproduzidos por eles com modificações para criar seus próprios jogos que seriam suas próprias simulações.

Ao definir o pensamento computacional como meta, estou valorizando padrões elevados de cognição, apontados pela taxonomia de Bloom. Valorizo o pensamento matemático e o pensamento complexo e oportunizo o desenvolvimento de competências como a abstração, a lógica, a criatividade, a autonomia e o trabalho colaborativo.

6.1.2 Aprendizagens sobre aprendizagem

O processo que estabeleci para proporcionar um ambiente de aprendizagem mais eficaz tem bases na literatura científica apoiando-se em três questões a que a investigação há muito procura responder e ainda não concluiu de forma indiscutível:

- Como é que as pessoas aprendem?
- Como é que se aprende matemática?
- Como é que se aprende Programação de Computadores?

Farei, por isso, uma breve alusão às respostas que encontrei.

O nosso conhecimento é construído de forma subjetiva através de modelos (Alves, 2000) ou metáforas (Postman, 2007). Os modelos ou metáforas se valem de símbolos culturalmente estabelecidos. É através desses sistemas de símbolos que as nossas mentes se comunicam com diferentes linguagens em diferentes mídias (Correia, 2003). Cada mídia fornece diferentes mediações que influenciam na nossa maneira de aprender e pensar (Borba, 2002; Tikhomirov,

1981). É através das narrativas que as pessoas criam, passam e repassam os conhecimentos, valendo-se das mídias e suas linguagens em um ambiente social, dentro de uma cultura criada historicamente. Pode-se tomar como unidade epistemológica do conhecimento os coletivos de seres humanos e suas mídias (Borba, 2002). Cada turma de Introdução à Lógica de Programação seria um coletivo que constrói o conhecimento ao produzir narrativas utilizando o computador e o Scratch como mídia. A narrativa aparece no produto final, em assuntos diversos relacionados à cultura dos jovens alunos. A narrativa aparece também no processo, quando eles debatem e se auxiliam mutuamente, falando sobre os problemas, técnicas e soluções adotadas em seus programas.

Para construir o conhecimento, estimula-se a construção de um artefato externo à mente humana, de modo a que se possa mostrar e debater com seus pares, o coletivo de seres humanos com mídias, testando, mostrando e debatendo o resultado dos seus pensamentos. De certa forma, depurando seus conhecimentos (Maltempi & Valente, 2000; Papert, 1980, 1997).

O Scratch é a ferramenta que possibilita a construção desses ativos externos mostráveis. No jardim de infância utilizam-se bolas, cubos, varetas, anéis, material de desenho, réguas, fitas, dobraduras e outros tantos materiais para que os alunos construam seus ativos externos como uma brincadeira que leva à experimentação e à construção do conhecimento (Arce, 2004; Kishimoto, 1996). O ambiente Scratch pode ser utilizado de forma análoga, sendo uma ferramenta tecnológica mais adaptada à faixa etária de jovens e adultos e mais em sintonia com nosso tempo tecnológico-digital (Resnick, 2007).

A criatividade pode ser estimulada. Para isso, alguns fatores favoráveis devem estar presentes, como Aptidões, Conhecimento, Motivação, Processos, Personalidade, Olhar do Outro. Os processos necessários à criatividade abrangem: flexibilidade de perceptiva, uso de imagens, pensamento analógico e metafórico, criação e descoberta de problemas e não apenas a sua resolução. Cabe ao professor valorizar estes requisitos da criatividade. Os fatores de personalidade mostram que a criatividade está associada à Autonomia, Autoconfiança, Abertura à experiência, Curiosidade, Sentido de humor, Tolerância à ambiguidade, Tomada de risco, Sensibilidade estética, Paixão pelo que se faz, Atração pela complexidade e Persistência. (Morais, 2012). Sendo assim, ao estimular a criatividade estimulam-se outras capacidades associadas, tais como a autonomia e o senso crítico. Por fim, o uso dessas competências estimula o pensamento computacional.

Atualmente os educadores matemáticos sugerem fortemente que se aprende matemática resolvendo problemas e refletindo sobre eles (Onuchic & Allevato, 2011; Smole & Diniz, 2001; Zulatto, 2007). Nesse processo de construção de conceitos se faz uso do raciocínio, da argumentação e da visualização.

As formas de raciocínio utilizadas pela matemática são a indução, a dedução e a abdução, sendo que o processo dedutivo se faz do geral para o particular, o processo indutivo das particularidades para a generalização e o abdutivo é uma combinação dos dois, acrescido da manipulação de conceitos de modo a testar conjecturas matematicamente (Zulatto, 2007). Por sua vez, a argumentação é a forma pela qual se atinge a visualização. Serve para convencer e criar os conceitos através de discussões no coletivo da classe, tal como fazem os matemáticos na comunidade científica. Estimular em sala de aula que os alunos pratiquem a matemática como fazem os matemáticos está de acordo com as ideias de Bruner quando afirma que cada aluno deve ser um minicientista. Nesse processo os alunos devem estabelecer algumas regras sociais e sociomatemáticas de modo a exercer a expressão audível, a escuta atenta, a partilha de ideias manifestando discordâncias justificadas. A prova matemática é um caso especial da argumentação na medida em que necessita da explicação do raciocino, formalmente até o produto final, enquanto a argumentação pode se valer de metáforas e analogias (Boavida, 2005; Zulatto, 2007). Por último, a visualização não se limita aos olhos, mas ao intelecto, podendo se utilizar de palavras e imagens para construir uma visão do conceito matemático nas mentes dos aprendizes (Zulatto, 2007).

Processos como abstração, decomposição e inferência caracterizam a matemática e fazem parte do pensamento computacional. A forma de trabalho, utilizando o convencimento e troca de experiências de forma colaborativa, também é valorizada como capacidade necessária ao pensamento computacional (National Research Council, 2010, 2011).

Nos anos de 1960 houve um movimento mundial conhecido como Matemática Moderna que, ainda hoje, influencia a forma como se concebe e se ensina a matemática. De acordo com essa corrente, matemática é um conhecimento pronto e formalizado, de verdades definitivas, infalíveis, imutáveis e sem uma conexão com a cultura. Usar o tempo para que o aluno recrie relações já conhecidas é frequentemente visto como desperdício. Uma vez que a matemática já está pronta e acabada, ao aluno cabe memorizar, assistindo passivamente as demonstrações, aplicar e reproduzir as soluções, entregando as respostas, rapidamente e sem erros. O

pensamento crítico e autônomo não é estimulado, favorecendo o autoritarismo e a submissão passiva e bloqueando a criatividade.

A partir dos anos de 1980, várias vertentes surgiram se opondo aos ideais da Matemática Moderna, conhecidas como Tendências em Educação Matemática. Como exemplo, pode-se citar as tendências da Resolução de Problemas, Etnomatemática, Modelagem e Tecnologias. De uma forma geral elas partem de uma vertente construtivista e interacionista, levando em consideração que o aluno aprende matemática dentro e fora da escola, dando abertura à cultura local como facilitadora do processo de aprendizagem. O professor passa a assumir a educação como um ato político, supondo que aprender matemática faz parte da educação crítica, passando valores e estimulando atitudes para modificar a realidade social. A matemática não é mais concebida como algo perfeito e acabado, mas sim como resultado de um processo histórico. Neste contexto o professor deve ser o professor-pesquisador, aprendendo e criticando suas próprias práticas à luz da realidade local e dos referenciais teóricos.

Ao se ensinar computação costuma-se ensinar os comandos e a sintaxe. No entanto, isto não é suficiente para ensinar programação. É como ensinar a acelerar, frear e girar o volante de um carro e supor que o aluno sabe dirigir (Pea & Kurland, 1984). Tem uma parte prática que leva o aluno a dominar o caráter sistêmico da programação, mas, para programar, o programador monta um modelo mental interno, descreve este modelo em uma linguagem de programação, executa o programa e reflete sobre os seus resultados. Na maioria das vezes estes resultados são erros ou simplesmente o programa não está como o programador deseja, levando-o a depurar o programa. Neste processo de descrição-execução-reflexão-depuração ele vai praticando e melhorando, na medida em que explora os recursos da linguagem (Maltempi & Valente, 2000). Este ciclo de descrição-execução-reflexão-depuração, correspondente à ideia de quebrar o problema em partes menores e menos complexas e a abstração de um modelo para manipular e encontrar uma solução, relaciona-se a competências da programação e do pensamento computacional que podem aplicar-se a outras áreas do conhecimento e da vida, se o ensino for voltado para esta extrapolação (Akcaoğlu, 2013).

A programação pode ser concretizada, a princípio, com qualquer linguagem, porém, o Scratch tem facilidades para iniciantes, como os encaixes dos comandos, que minimizam os erros de sintaxe. O Scratch funciona segundo uma metáfora de Palco ao fundo e Objetos (ou Atores) que interagem. Neste micromundo, os alunos são capazes de fazer jogos e animações e,

ao construir estes micromundos, os alunos constroem o seu próprio conhecimento. Durante a investigação pude constatar esta construção do conhecimento quando os alunos, depois de experimentar, explicam aos colegas o funcionamento dos programas ou quando ajudam um colega a achar um defeito. Os programas refletem os modelos mentais dos alunos, que podem ser avaliados, criticados e melhorados, levando o aluno a refazer seus modelos mentais. Os projetos de programação são recomendados para que o aluno exercite o construcionismo com criatividade. Nessa convicção, menciono aqui alguns motivos para utilizar projetos nas aulas de programação, sugeridos por Resnick (2007):

- Requerem mais envolvimento do aluno;
- Costumam ser multidisciplinares;
- Evitam o pensamento dicotômico (certo/errado);
- Os produtos construídos servem de reflexão, ajustando os modelos mentais internos de funcionamento do mundo;
- Provocam empatia, uma vez que os alunos precisam considerar o olhar do outro sobre seu produto construído.

6.1.3 Aprendizagem: um caminho com muitas etapas

Com base nas leituras sobre aprendizagem e a prática através dos anos de 2011 a 2014, desenhei um processo de ensino de programação para melhorar a aprendizagem de meus alunos, futuros professores de matemática, com a intenção de incentivar as competências relacionadas ao pensamento computacional.

O ambiente de aprendizagem aqui apresentado, guiado por Roteiros, Prova e Projetos, foi resultado de um processo histórico exposto na autobiografia. Os projetos resultantes do processo de aprendizagem proposto mostraram evidencias do pensamento computacional, segundo a ferramenta Dr. Scratch. Nas atividades de realização do projeto os alunos devem resolver problemas, conceber produtos e informações e tomar decisões, evidenciando assim o pensamento complexo.

A análise documental mostrou-se consistente com os resultados da atuto-boiografia, usando a taxonomia de Bloom para uma avaliação cognitiva. A taxonomia de Bloom pode ser mapeada para as capacidades do pensamento computacional através da análise apresentada por Cynthia Selby (Selby, 2014), ao atingir os níveis mais altos da taxonomia. O Mapeamento

das competências de pensamento computacional e os níveis mais altos da taxonomia de Bloom atualizada em 2001 estão representados na Tabela 41.

Tabela 41 - Mapeamento da Taxonomia de Bloom vs Pensamento Computacional

Processos Cognitivos da Taxonomia de Bloom resta em 2001	Competências do Pensamento Computacional
Analisar	Abstração de funções Abstração de dados Decomposição
Avaliar	Avaliação
Criar	Projeto de algoritmo

Considerando o mapeamento de Selby (Tabela 41), notamos que os projetos são as atividades que mais contribuem para o pensamento computacional. Selby afirma que as capacidades relativas ao pensamento computacional referentes a Análise (Abstração e Decomposição), são as mais difíceis de alcançar pelos alunos iniciantes. Esta afirmação confirma as dificuldades que meus alunos sentem para realizar a prova.

6.1.4 Aprendizagem: um caminho com dois sentidos

Neste item comento o que aprendi ao realizar a pesquisa sobre o processo de aprendizagem de programação, que é o objeto dessa investigação. Percebo que muitas das contribuições apresentadas se devem aos erros que cometi. Os erros e acertos me conduziram à reflexão sobre o processo enquanto ensinava, caracterizando um aprendizado com dois sentidos. Espero que o conhecimento que adquii também seja útil a outros professores de programação e de matemática, assim como a outros investigadores interessados no tema, ao refazer o processo ensinando e aprendendo com ele.

Neste trabalho defini um processo para o ensino de programação, promovendo as capacidades relacionadas com o pensamento computacional que considero bem-sucedido. O processo aqui estabelecido não é uma receita de sucesso, mas é mais uma opção de que o professor pode lançar mão. Os roteiros, enquanto tarefas que permitem a intervenção do professor, escolhendo os exemplos e os conteúdos a serem ensinados, têm um caráter mais

diretivo e amparado de ensino e aprendizagem, enquanto os projetos, mais a cargo dos alunos, permitem a construção de conhecimento, o desenvolvimento da autonomia e a expressão criativa.

Nas provas passei a solicitar a explicação de dois pequenos programas e esta forma de avaliar se mostrou melhor do que a solicitação de algoritmos escritos, uma vez que, desenvolver programas usando a mídia de lápis e papel, não se tem o retorno proporcionado pelo computador. Ao descrever os programas os alunos revelam os conceitos adquiridos, dando a oportunidade do professor detectar conceitos errados e intervir imediatamente.

O *storyboard* se mostrou adequado ao planejamento do projeto, uma vez que, com um desenho para cada cena, é possível identificar os objetos e as interações entre eles. Também foi útil para o acompanhamento do projeto final e na discussão com os alunos, pois reconhecemos as cenas mais fáceis e as mais difíceis. Com duas semanas para a execução do projeto, tentámos balancear a carga de trabalho Após a primeira semana, reavaliámos e refizemos as metas para a semana seguinte, baseando as discussões e reavaliações no *storyboard*.

As narrativas aparecem como um instrumento de ensino. Elas envolvem os alunos, como no caso dos roteiros, onde utilizei nomes como "Gato Caminha", "Voo do Morcego" ou "Andamento do Ciclista", para passar os conceitos relacionados com Contador, Função Senoidal e Estruturas de Programação. O conteúdo a ser ensinado está embalado pela narrativa e quando preciso de um contador eu falo "Vamos fazer um contador. Lembra? Como no Gato Caminha". Cada programa tem uma pequena narrativa, um pequeno imaginário. Os projetos de animações e os jogos trazem uma narrativa, fazendo da programação com Scratch um meio de expressão.

Outro aprendizado é que os jogos eletrônicos fazem parte da cultura dos alunos e, por isso, podem despertar o interesse pela programação. Em 2013 investi muito nesse aspecto e notei que eles conseguiram fazer roteiros bem complexos, imitando jogos como Space Invader ou Wumpus, mas estes roteiros fizeram com que os alunos se aprofundassem em atividades de implementação, característica do nível cognitivo 3 na taxonomia de Bloom (Aplicar), sem tempo para os alunos personalizarem os jogos exercitando a imaginação, característica do nível cognitivo 6 na taxonomia (Criar). Na minha avaliação, o ganho em técnica prejudicou a criatividade que eu buscava. Sendo assim, reduzi a complexidade dos jogos para recuperar o trabalho com a criatividade.

6.1.5 Scratch, criatividade e influência social

O Scratch provou ser uma ferramenta fundamental nesse processo de aprendizagem. O fato de os programas serem gerados com comandos arrastados, sem a necessidade de digitação, eliminou os problemas de sintaxe comuns em linguagens textuais. A possibilidade de se utilizarem figuras e sons provoca uma simpatia pelos programas criados. A facilidade de se criarem animações motiva os alunos e gera um retorno (*feedback*) imediato dos seus trabalhos, que são consequências de seus modelos cognitivos. Conceitos como processamento paralelo e orientação a objetos são facilmente implementados. As manifestações expressas por Mitchel Resnick sobre aprendizagem no estilo jardim de infância são percebidas, ou seja, o Scratch foi criado para que as pessoas continuem experimentando, conjecturando e se divertindo no processo de aprender, como no jardim de infância (Resnick, 2007).

A sequência dos projetos de animação e jogo teve um padrão de crescentes complexidade e criatividade. O fato de propor um pequeno projeto antes do projeto final resultou num aprendizado vindo da experiência em 2011, quando os meus alunos pularam da fase de roteiros para um projeto final e não estavam acostumados a criar suas próprias produções. Realmente eles passaram por um bloqueio que não aconteceu no ano seguinte, quando coloquei um projeto de animação antes do projeto final. Esta foi mais uma lição aprendida que merece ser registrada. Esta dificuldade que meus alunos enfrentam em serem criativos também aparece no estudo de Luiz Valente ao ministrar um curso de programação de computadores para turmas de licenciatura em educação básica em Portugal.

> Mesmo quando eram incentivados a esquecerem-se dos exemplos de materiais com que estavam habituados a contactar para que arriscassem a ser inovadores, ousados, distintos, aconselhados por mim e pelo docente responsável pela unidade curricular, revelavam-se incapazes de ser originais e resistiam às nossas propostas com manifestações de visível desconforto. (Valente, 2011, p. 151)

Maltempi e Valente dão uma explicação sobre o fato que me parece plausível, ao afirmar que a escola privilegia a memorização e a reprodução de informações e pune os erros (Maltempi & Valente, 2000). Isso leva os alunos a ficarem apáticos se não sabem a resposta certa. Em um projeto não existe resposta certa, apesar de existirem várias possibilidades. Nesse caso os alunos precisam ser incentivados a arriscar.

Para enfrentar esta situação, comecei sugerindo que modificassem os roteiros, personalizando os programas, e introduzi o projeto de animação antes do projeto final. O projeto

de animação, com um tema comum a toda classe, ajuda os alunos a terem ideias interagindo uns com os outros. No projeto final, eles ficam mais confiantes e levam adiante um tema e uma história original. Esta criatividade também ajuda a resolver problemas, na medida em que eles arriscam soluções, dando à criatividade um caráter utilitário e emancipatório (Morais, 2012).

Ao utilizar a criatividade o aluno também desenvolve a expressão e a socialização, considerando o outro como juiz do seu desempenho. A comunicação e a estética passam a ser importantes para o seu trabalho.

O processo de aprendizagem começa com os roteiros e finaliza com os projetos. Neste percurso o protagonismo do professor diminui, à medida que o do aluno aumenta. As tarefas propostas nos roteiros e os vídeos gravados com aulas expositivas retiram do professor o papel de reprodutor de informações em sala de aula e aumenta seu trabalho com planejamento e reflexão sobre os resultados. O professor deve utilizar as aulas para estar próximo e atento aos alunos, observando suas dificuldades e os seus interesses. O professor também deve atuar para mudar a concepção de que o bom aluno é o que fica calado, trabalha sozinho e tem as respostas prontas. O protagonismo, a argumentação, a reflexão e a socialização devem ser incentivados para atingir as metas estabelecidas pelo professor. A atitude do professor e a consciência do seu papel são importantes para que o processo de aprendizagem seja bem sucedido. Em se tratando de um curso de formação de professores, estas concepções devem ser explicitadas com a intenção de que os futuros professores adotem esta postura.

A teoria sócio-interacionista destaca a importância da interação com o meio e com os outros indivíduos e a força da cultura vigente na sociedade. O professor se utiliza de palavras, signos e sentidos para que o aluno signifique e ressignifique seus conceitos e o os conteúdos programáticos exigidos socialmente através da escola (Barros, Paula, Pascual, Colaço, & Ximenes, 2009). Neste trabalho procurei contar com a força dessa coletividade para me ajudar e para ajudar os alunos no processo de aprendizagem. A experiência demonstrou que, somente eu não sou capaz de resolver todas as dúvidas que aparecem nos procedimentos e que a ajuda de outros alunos é fundamental para o bom andamento da aula. Este contato entre eles ajuda a criar laços de confiança e amizade. Quero lembrar que os alunos são calouros e que a maioria está se conhecendo neste semestre.

A colaboração, desde os primeiros roteiros, promove a socialização e cria laços afetivos. A personalização dos programas promove uma identificação com o objeto criado, provocando a

curiosidade de seus colegas. Os colegas são necessários como plateia e colaboradores do seu trabalho, tal como identificaram Valente e Osório (2009). Nesse processo o aluno cria sua identidade na medida em que se integra ao grupo. Os projetos de animações com um tema comum, produzindo um vídeo com contribuições de toda a classe, reforçam estes laços de cumplicidade. No projeto final é rotineira a ajuda entre os grupos. A colaboração do outro dá mais confiança para que os alunos se disponham a resolver os desafios da programação, facilitando o caminho pela zona de desenvolvimento proximal rumo à autonomia. A perspectiva do outro se mostra importante para desenvolver a técnica, a criatividade e a colaboração.

A apresentação final é um momento especialmente importante, porque é quando os alunos mostram e assistem às apresentações dos projetos dos outros. Eles valorizam e se interessam pelos projetos dos seus colegas. Reconhecem as dificuldades encontradas, têm oportunidade para agradecimentos a alunos de outras equipes que os ajudaram.

6.2 Informações e indicações para adaptar e reutilizar o processo

Neste parte destaco o "como fiz", na esperança de revelar onde se pode adaptar, caracterizando uma contribuição de ordem prática e pragmática na adoção do processo de aprendizagem proposto. As animações e jogos são os mais variados, dependendo da vontade dos alunos. Os mecanismos de jogos mais procurados são do tipo "Plataforma" e "Tiro em Primeira Pessoa". Jogos de plataforma são do tipo "corre e pula", como o Mário Bros. Jogos do tipo "Tiro em Primeira Pessoa" (FPS – First Person Shooter) são aqueles em que a tela mostra a visão do atirador para acertar nos alvos na tela. Estes dois mecanismos de jogo foram desenvolvidos, pela primeira vez, pelos alunos, e só posteriormente eu os modifiquei para serem utilizados como modelos dos jogos.

A maior intervenção que os professores podem fazer é nos roteiros, ao definir os exemplos e conteúdos básicos que desejem propor nesta fase propedêutica. Mas, mesmo nessa fase, os roteiros se preocupam com o que deve ser feito, enquanto a manipulação e exploração da ferramenta Scratch fica a cargo dos alunos. O fato de os comandos terem uma lógica de agrupamento e saber como manipular os blocos, aprende-se por tentativa e erro, comprovando que existem fatos e processos que não é preciso se ensinar. Ao estabelecer o objetivo o aluno

supera as dificuldades como na experiência *The Hole in The Wall* conduzida por Sugata Mitra (2000).

No estudo de caso apresento os roteiros e a intenção didática de cada um deles. O professor que quiser repetir esta experiência, adaptando-a ao seu contexto, tem nesse material um ponto de partida, reproduzindo, testando, questionando e modificando, não necessariamente nesta ordem. Os materiais utilizados se encontram no link http://goo.gl/xQHHQP .

Outra experiência que merece destaque é a prática da Sala de Aula Invertida. As explicações que antes eu fazia sobre os roteiros e as teorias envolvidas nas tarefas, eu passei a registrar em vídeo. Assim, restou mais tempo em sala de aula para eu acompanhar as atividades. Os alunos gostaram da experiência dizendo que eles poderiam ver e rever as aulas no ritmo e hora que fossem convenientes. As aulas em vídeo ministradas em 2014 estão disponíveis em http://goo.gl/ElUta0. A intenção da Sala de Aula Invertida não é se valer de uma aula pronta, disponível na Internet e utilizá-la, mas sim que o professor produza um conteúdo de acordo com as necessidades e perfil da classe e da turma específica, que os alunos devem explorar em seu próprio ritmo, em casa ou, pelo menos, fora da escola, aproveitando para colocar as dúvidas na sala de aula.

Storyboard serve como documento de planejamento e dá suporte ao acompanhamento dos projetos. Solicitei um *storyboard* para o projeto de animação e outro para o projeto final. A experiência de realizar o primeiro, ajuda na elaboração do segundo. Com minha experiência estabeleci algumas dicas para o desenvolvimento:

1. Escolha o tema;
2. Discuta em linhas gerais como será a história;
3. Desenvolva um *storyboard* com um quadro para cada cena.

As provas se mostram um instrumento interessante, na medida em que, ao descreverem os programas, os alunos revelam suas concepções e conceitos. Para descrever um programa considero importante que se faça da seguinte forma:

1. Descrever o programa de forma geral do ponto de vista do usuário.
2. Mencionar os objetos que interagem entre si.
3. Descrever os blocos de código de cada objeto, relacionando-os com os efeitos para o usuário.

Para chegar à prova utilizei duas aulas antes como preparação na seguinte sequência:

- Prova Parte 1: Aula com dois programas simples, na qual os alunos vão explicando os programas de forma coletiva e eu vou tentando melhorar as frases pronunciadas, de forma que fiquem mais claras e utilizem um vocabulário adequado. Depois peço para que escrevam a explicação do segundo programa. Após a escrita peço que alguns voluntários leiam o que escreveram e faço comentários sobre as explicações que os alunos dão. Eles vão para casa com a tarefa de explicar outros dois pequenos programas.
- Prova Parte 2: As explicações feitas em casa são trocadas entre os alunos e peço para que avaliem as explicações uns dos outros. Depois eles repassam ao avaliado as suas impressões. No final da aula costumo perguntar o que eles acharam da atividade, se explicar e avaliar é fácil ou difícil e, por que.
- Prova Parte 3: Nesta aula costumo aplicar a prova que será corrigida com o mesmo critério de avaliação que eles utilizaram.

Para avaliar as provas faço da mesma forma que oriento os alunos na Prova Parte 2. Peço que eles assinalem três tipos de frases marcando com C, X e O.

- C: Certo. Frases que mostram causa no código e efeito na interface do usuário.
- X: Errado. Frases com erros de interpretação ou erros conceituais.
- O: Nada. Frases que não ficaram claras, ou cujo significado o avaliador não conseguiu entender.

A marcação dos três tipos de frases caracteriza uma análise de detalhes. Estas marcações servem de base para justificar um comentário geral sobre o desempenho do avaliado. Depois é feita uma classificação entre cinco valores: Muito Ruim, Ruim, Regular, Bom, Muito Bom.

Depois de várias tentativas, concluí que o projeto final deve ser feito em duplas e a métrica LOC é adequada, quando a tomamos em escala logarítmica, ao número de comandos utilizados, à avaliação do desempenho dos alunos. Da mesma forma, a métrica da complexidade ciclomática de McCabe também pode ser utilizada para medir a complexidade do desempenho.

Alguns conteúdos, cujos erros conceituais são recorrentes, podem ser reforçados por aulas específicas, de forma presencial ou em vídeo. No meu caso identifiquei que o algoritmo que caracteriza os contadores e as estruturas básicas de programação, diferenciando condicional de loops, deveria ser reforçado.

6.2.1 Outras Ferramentas

Antes de descobrir o Scratch, fiquei pesquisando ferramentas que possibilitassem o uso de animações, jogos e orientação a objetos na programação. Pesquisei bibliotecas em C++ e a linguagem Lua, com biblioteca de desenvolvimento de jogos, mas elas não eram de uso imediato e, por vezes, pensei que estava complicando ao invés de simplificar. Dessa maneira reconheço a importância das ferramentas que viabilizam projetos mais consonantes com o pensamento computacional. O pensamento computacional representa as competências necessárias para a sociedade resolver as questões com que se depara no século XXI. Ferramentas como estas são necessárias para facilitar o aprendizado compatível com nosso tempo, como constatou Alan Kay e Mitchel Resnick (Kay, 2003, 2007; Resnick, 1998, 2002, 2007). Por essas razões, penso que algumas das investigações futuras devem se basear em ferramentas que proporcionem um ambiente favorável ao desenvolvimento do pensamento computacional.

Uma reinvindicação recorrente dos alunos é a de fazer programas mais complexos, com a qualidade semelhante aos que eles estão acostumados a jogar, tendo o mercado de jogos profissionais como referência. Com este intuito, pretendo investigar a ferramenta Stencyl, identificada como alternativa pelo grupo de desenvolvimento do Scratch no MIT Media Lab. Esta ferramenta também usa os comandos em blocos de encaixe, evitando os erros de sintaxe e é possível a realização de jogos mais complexos com simulações das leis da Física, sendo capaz de criar jogos para celular, que são outra reinvindicação frequente entre os alunos. Para usar o Stencyl e publicar na web, a versão é gratuita.

Outra ferramenta interessante é o RPG Maker. Embora não exija programação, um projeto de RPG demanda planejamento, criação e resolução de problemas, desde a sua narrativa até a contagem de pontos e avaliação de probabilidades de sucesso do seu herói. No jogo de RPG um herói é encarregado de uma missão e enfrenta desafios. Do ponto de vista da educação, podemos aproveitar estes desafios como estímulo ao aprendizado de matemática ou de qualquer disciplina. Os níveis dos jogos podem representar patamares de conhecimentos (*scaffolds*) a serem alcançados. Podem-se utilizar os RPGs como mais uma mídia de ensino, numa perspectiva instrucionista, ou como um projeto a ser construído pelo aluno, numa perspectiva construcionista. Porém, esta ferramenta não é gratuita, embora existam opções alternativas como RPG Boss (http://rpgboss.com) que é uma ferramenta livre e de código aberto.

6.3 Investigação futura

Exponho aqui minhas ideias para futuras investigações com a intenção de realizá-las e inspirar outros professores e investigadores em trabalhos futuros.

A ferramenta Scratch possibilitou o processo de aprendizagem adotado. No entanto, pouco material didático se encontra em língua portuguesa. Uma das perspectivas para trabalhos futuros é a produção de material em Português. Penso em produzir material didático para ensinar a programar, incluindo a intenção de possibilitar ao aluno ser criativo e autônomo, usando as competências do pensamento computacional como política a ser valorizada. O incentivo à criação de projetos é a maneira que percebo de incentivar o pensamento computacional e a interdisciplinaridade.

Os roteiros são a porta de entrada do processo, a fase onde o professor é mais proativo que o aluno. Neste trabalho os roteiros privilegiam relações com a matemática do ensino médio, mas outras possibilidades podem ser abertas relacionando os roteiros em Scratch, com álgebra, vetores, geometria analítica e teoria dos conjuntos. Esta perspectiva para os roteiros atenderia aos alunos que estão há mais tempo cursando matemática e aos repetentes das disciplinas. Outra perspectiva para os roteiros é a computação, explorando técnicas como diagramas de estado, a metáfora de objetos e algoritmos tradicionais. Professores ligados às letras podem desenvolver roteiros explorando as formas de expressão, narrativas de animações e jogos, numa perspectiva semiótica. As perspectivas podem ser ampliadas horizontalmente, na medida em que mais perspectivas podem ser incorporadas e verticalmente, na medida em que mais roteiros podem ser acrescentados a cada uma das perspectivas. Estas perspectivas não são antagônicas e sim complementares. Sendo assim, estes roteiros, com várias perspectivas, podem se tornar portas de entrada para a aprendizagem de programação passíveis de serem negociadas entre alunos e professores. Com a programação e os projetos presentes, todas estas vertentes terão, no final, contribuído para o pensamento computacional.

Outra possibilidade de trabalhos futuros é a adoção de projetos colaborativos, nos quais alunos devem se organizar para trabalharem em um único projeto. Esta forma de trabalho, na qual os projetos são repartidos em pequenas unidades que se relacionam, faz surgir desafios de planejamento, sincronismo e comunicação em níveis mais elevados do que os projetos concebidos em um único bloco. Este tema merece investigação, seguindo a vertente que

concebe pensamento computacional como Forma de Colaboração (National Research Council, 2011).

6.4 Conclusão

Apesar de a literatura acadêmica valorizar a criatividade e a autonomia, não encontramos muitos trabalhos académicos que mostrem de forma pragmática como começar a incentivar estas competências. Pude constatar este quadro ao analizar os trabalhos acadêmicos relacioandos com o meu nos últimos 5 anos, pesquisados nos repositórios Open Access Infrastructure for Research in Europe (OpenAIRE), Repositório Científico de Acesso Aberto de Portugal (RCAAP), RECOLECTA (Recolector de Ciencia Abierta), Networked Digital Library of Theses and Dissertations (NDLTD), Banco de Teses da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes) O professor, como último indivíduo da estrutura hierárquica educacional, atua diretamente na base da pirâmide do sistema de ensino onde estão os alunos. Cabe ao professor realizar na prática o que foi idealizado pelo topo da estrutura como os ministérios da educação de seus países ou mesmo órgãos internacionais como a UNESCO. Na distância entre o topo e a base da pirâmide aparecem fatores políticos, sociais e econômicos, assim como as concepções hegemônicas locais sobre a educação. O professor, por vezes, deve adaptar suas práticas para conseguir objetivos claros e realizáveis. Mas ele pode não saber que práticas adotar para chegar aos seus objetivos. Neste trabalho encontra-se um modelo baseado em roteiros, provas e projetos que pode ser usado e adaptado, auxiliando o professor a conciliar o idealizado pelo topo as práticas da base da pirâmide, sem restringir suas opções. O professor deve ser livre para ajustar aos seus alunos e à cultura local. A flexibilidade é necessária uma vez que cada classe tem sua própria identidade, seus próprios interesses e sua maneira de ver o mundo. Este trabalho tem a intenção de ajudar professores que querem mudar as suas práticas assim como aqueles que vão dar o primeiro passo.

Os roteiros são procedimentos que levam à realização de pequenos programas. Cada roteiro traz a pequena satisfação de um trabalho realizado, dando ao aluno a possibilidade de adaptação e personalização. Este é o modelo que utilizei para colocar o conteúdo da matemática escolar enquanto ensinava a programação, mas o modelo está aberto à intervenção do professor podendo ele criar seus próprios roteiros com a vertente que desejar nesta fase inicial na qual os primeiros conceitos de programação são aprendidos. A ideia das provas descritivas veio de

atividades de matemática na qual o aluno explicava o seu raciocínio que o levou a resolver os problemas propostos. Ao explicar os programas, os alunos têm que relacionar as partes codificadas com as técnicas e procedimentos utilizados e imaginar como vai funcionar o programa, isto é, quais as consequências ao submeter aquela descrição à máquina de inferências do computador. Ao avaliar, com a prova descritiva, a avaliação do pensamento computacional é intermediada pela competência em elaborar o texto, ou seja é, as vezes o programa é compreendido mas o aluno não consegue se expressar por escrito influenciando na avaliação. Na análise do texto escrito, pude observar erros conceituais como a diferenciação entre condicional e *loop*. Outra opção é a elaboração de uma prova de múltipla escolha, retirando a elaboração do texto como distorção da avaliação, mas também retira o treinamento da elaboração de texto. Nos projetos os alunos mostram criatividade usando o Scratch como forma de expressão. Aprendi que o projeto de animação é um degrau necessário para um projeto maior. O tema em comum da animação para toda a classe estimula as conversas e a troca de ideias entre os alunos, criando uma confiança que se reflete no projeto maior com os temas individualizados. Para os projetos se faz necessário conceber ideias, tomar decisões e resolver problemas, o que caracteriza o pensamento complexo definido pelo modelo em *A Guide to Developing Higher Order Thinking Across the Curriculum* (Burklund, 1989). Uma análise com a ferramenta Dr. Scratch, em uma amostra dos programas desenvolvidos pelos alunos, evidenciou um nível elevado do pensamento computacional.

Usando a taxonomia de Bloom como referência, pode-se observar que Roteiros, Prova e Projetos, nesta sequência, são atividades que atuam de modo a complementar as dimensões cognitivas e de conhecimento de forma crescente. Os Roteiros atuam predominantemente nos níveis mais baixos da dimensão cognitiva (Lembrar, Entender, Aplicar) e da dimensão do conhecimento (Factual, Conceitual e Processual). A Prova atua predominantemente nos níveis intermediários da dimensão cognitiva (Analisar) e na dimensão do conhecimento (Conceitual e Processual). Os Projetos atuam nos níveis mais elevados na dimensão cognitiva (Avaliar e Criar) e na dimensão do conhecimento (Conceitual, Processual e Metacognição). Os projetos, ao atingir as regiões mais elevadas da taxonomia de Bloom, também evidenciam o uso do pensamento computacional segundo o estudo de Selby (2014). O *storyboard* se mostrou útil tanto para o planejamento quanto para o controle do projeto. Ele deu a oportunidade da criação independente da programação e também ajudou na metacognição, quando o aluno avalia seu

próprio trabalho ajustando o projeto as restrições, facilidades, prazos e suas próprias competências.

O pensamento computacional aparece com naturalidade em projetos de desenvolvimento de software e este estudo mostrou competências relevantes e comuns com o pensamento matemático, inclusive quando se trata da valorização do trabalho colaborativo e de uma preocupação com autonomia e criatividade como fatores socialmente importantes. A abstração, simulação, modelagem, articulação lógica, criação de algoritmos, pensamentos sistêmicos, resolução de problemas, articulação de símbolos e identificação de padrões são competências comuns ao pensamento matemático e ao pensamento computacional. A consciência do pensamento computacional e a competência de programar computadores podem ser motivadores para ensinar matemática usando ferramentas como Scratch para criar modelos, conjecturas e experimentar a matemática. A interação social é desejada e trabalha a favor do ambiente de aprendizado. Os alunos têm em seus colegas auxílio e plateia, incentivando a realização de seus programas. Jogos e animações criaram uma empatia cultural com estes jovens identificados como nativos digitais. A narrativa foi mais uma estratégia para envolver os alunos de forma afetiva. Os roteiros possuíam suas narrativas embalando o conteúdo ensinado e, ao criar suas narrativas, alterando os roteiros e criando projetos os alunos intensificavam suas relações sociais.

Roteiros, Prova e Projetos são instrumentos que conduzem o processo e foram construídos e utilizados com a concepção de que a aprendizagem se faz pela interação social e com a crença de que criatividade e autonomia devem ser valorizadas. Espero que estes valores e concepções possam influenciar outros professores e também meus próprios alunos que se tornarão professores, já que este trabalho se trata de uma maneira de “como fazer” que desperta possibilidades de alterações e adaptações. Sendo assim entendo ter satisfeito a questão de investigação.

7 Referências

Akcaoğlu, M. (2013). *Cognitive and motivational impacts of learning game design on middle school children.* (Ph. D. thesis), Michigan State University, East Lansing, MI.

Aldana-Avilés, D. L. (2015). El lenguaje de programación Scratch como material didáctico motivador para la aplicación y evaluación de contenidos en el área de inglés para alumnos con diagnóstico de TDAH.

Alencar, G. A., Freitas, A. K., & Danielle, J. S. (2014). Utilizando o Scratch nas aulas de Lógica de Programação do Proeja: Um relato de experiência. In J. Sánchez (Ed.), *Nuevas Ideas en Informática Educativa: Memorias del XIX Congreso Internacional Informática Educativa, TISE 2014, Fortaleza, Brasil* (pp. 542-545): Universidade Federal do Ceará.

Almeida, F. C. (2011). O historiador e as fontes digitais: uma visão acerca da internet como fonte primária para pesquisas históricas. *AEDOS, 3*(8).

Alves, R. (2000). *Filosofia da ciência: Introdução ao jogo e a suas regras* (Vol. 8). São Paulo, Brasil: Edições Loyola.

Anderson, L. W., & Krathwohl, D. R. (Eds.). (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. New York, NY: Longman.

Arce, A. (2004). O jogo e o desenvolvimento infantil na teoria da atividade e no pensamento educacional de Friedrich Fröebel. *Caderno Cedes, Campinas, 24*(62), 9-25.

Aureliano, V. C. O., & Tedesco, P. C. A. R. (2012). Avaliando o uso do Scratch como abordagem alternativa para o processo de ensino-aprendizagem de programação. In L. C. P. Albini, C. Menegazzo & C. R. Nichele (Eds.), *Anais do CSBC 2012 - XXXII Congresso da Sociedade Brasileira de Computação* (Vol. 32, pp. 1-10). Curitiba, Brasil: Sociedade Brasileira de Computação.

Balanskat, A., & Engelhart, K. (2014). Computing our future: Computer programming and coding-Priorities, school curricula and initiatives across Europe. European Schoolnet. Acedido em 27 de fevereiro, 2015.

Barcelos, T. S., & Silveira, I. F. (2012). *Pensamento Computacional e Educação Matemática: Relações para o Ensino de Computação na Educação Básica.* Paper presented at the XX Workshop sobre Educação em Computação.

Barreto, A. L. O., Camelo, L. S., Fernandes, A. C., Pequeno, M. C., & Filho, J. A. C. (2009). Investigando a Contribuição do Software Educativo Winplot para a Compreensão do Conceito de Função. In R. A. Silveira (Ed.), *Anais do XX Simpósio Brasileiro de Informática na Educação (SBIE 2009), Florianópolis, SC, Brasil*.

Barros, J. P. P., Paula, L. R. C., Pascual, J. G., Colaço, V. F. R., & Ximenes, V. M. (2009). O conceito de "sentido" em Vygotsky: considerações epistemológicas e suas implicações para a investigação psicológica. *Psicologia & Sociedade, 21*(2), 174-181. doi: 10.1590/S0102-71822009000200004

Bezerra, L. N. M., & Silveira, I. F. (2011). *Licenciatura em Computação no Estado de São Paulo: uma Análise Contextualizada e um Estudo de Caso.* Paper presented at the Proc. XXXI Congresso da Sociedade Brasileira de Computação-IX Workshop de Educação em Computação, Natal, RN, Brasil.

Biggs, J. (2003). Aligning teaching and assessing to course objectives. *Teaching and Learning in Higher Education: New Trends and Innovations. University of Aveiro, 13-17 April, 2003*. Aveiro. Portugal: University of Aveiro.

Biggs, J., & Tang, C. (2011). *Teaching for quality learning at university* (4th ed.). Maidenhead, UK: Open University Press.

Bitoun, J. (2005). O que revelam os índices de desenvolvimento humano. *Atlas do Desenvolvimento Humano no Recife* (pp. 1-59). Recife, Brasil: Prefeitura do Recife.

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). *Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain*. New York: David McKay Company.

Boavida, A. M. R. (2005). *A argumentação em Matemática: Investigando o trabalho de duas professoras em contexto de colaboração.* Universidade de Lisboa, Lisboa.

Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013). *Hairball: Lint-inspired static analysis of scratch projects.* Paper presented at the Proceeding of the 44th ACM technical symposium on Computer science education.

Borba, M. C. (2002). Coletivos seres-humanos-com-mídias e a produção de Matemática. In Universidade Federal do Pará, Universidade Tuiuti do Pará & Pontifícia Universidade Católica do Pará (Eds.), *Anais: Trabalhos completos: I Simpósio Brasileiro de Psicologia da Educação Matemática* (pp. 135-146). Curitiba, Brasil: Sociedade Brasileira de Psicologia da Educação Matemática & Sociedade Brasileira de Educação Matemática.

Borba, M. C. (2007). Humans with Media: a performance collective in the classroom? In G. Gadanidis & C. Hoogland (Eds.), *Proceedings of the fields institute digital mathematical performance symposium* (pp. 15-21). Toronto, Canada: Fields Institute for Research in Mathematical Sciences.

Borba, M. C. (2012). Humans-with-media and continuing education for mathematics teachers in online environments. *ZDM Mathematics Education, 44*, 801-814. doi: 10.1007/s11858-012-0436-8

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking.* Paper presented at the Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada.

Bruner, J. (2001). *A cultura da educação*. Porto Alegre, Brasil: Artmed.

Bueno, B. O., Chamlian, H. C., Sousa, C. P. d., & Catani, D. B. (2006). Histórias de vida e autobiografias na formação de professores e profissão docente (Brasil, 1985-2003). *Educação e Pesquisa, 32*(2), 385-410.

Burklund, C. (1989). A guide to developing higher order thinking across the curriculum. Des Moines, IA: Iowa State Department of Education.

Coll, C., Martín, E., Mauri, T., Miras, M., Onrubia, J., Solé, I., & Zabala, A. (2001). *O construtivismo na sala de aula: Novas perspectivas para a acção pedagógica*. Porto: Edições Asa.

Coordenadoria de Ensino e Integração Acadêmica, Núcleo Didático-Pedagógico. (2007). *Projeto Pedagógico do Curso de Licenciatura em Matemática Diurno*. [Programa do curso]. Centro de Formação de Professores, Recôncavo da Bahia, Brasil.

Correia, M. (2003). A constituição social da mente: (re)descobrindo Jerome Bruner e construção de significados. *Estudos de Psicologia, 8*(3), 505-513.

Correia, T. F. M. (2013). *Scratch na aprendizagem da matemática.* (Master thesis), Instituto Politécnico de Setúbal.

Coutinho, C. P. (2013). *Metodologia de investigação em Ciências Sociais e Humanas: Teoria e prática* (1.ª ed.). Coimbra, Portugal: Almedina.

Coutinho, C. P., Sousa, A., Dias, A., Bessa, F., Ferreira, M. J. R. C., & Vieira, S. R. (2009). Investigação-acção: metodologia preferencial nas práticas educativas. *Revista Psicologia, Educação e Cultura, 13*(2), 355-379.

D'Ambrosio, U. (1996). *Educação Matemática: da teoria à prática*: Papirus Editora.

Dalla Vecchia, R. (2012). *A modelagem matemática e a realidade do mundo cibernético.* (Tese de Doutoramento), Universidade Estadual Paulista, Rio Claro, São Paulo, Brasil. Retrieved from http://www.rc.unesp.br/gpimem/downloads/teses/dalla_vecchia_r_dr_rcla.pdf

Denning, P. J. (2009). The profession of IT Beyond computational thinking. *Communications of the ACM, 52*(6), 28-30.

Denning, P. J. (2010). Ubiquity symposium'What is computation?': Opening statement. *Ubiquity, 2010*(November), 1.

Dias, A. L. B. (2008). Resolução de problemas. In C. A. Muniz (Ed.), Programa Gestão da Aprendizagem Escolar - Gestar II. Matemática: Caderno de Teoria e Prática 1 - TP1: matemática na alimentação e nos impostos (pp. 45-54). Brasília: Ministério da Educação, Secretaria de Educação Básica. Retrieved from http://portal.mec.gov.br/seb/arquivos/pdf/2008/gestar2/matematica/tp1_matematica.pdf.

Dias, K. L., & Serrão, M. d. L. (2014). A linguagem Scratch no ensino de Programação: Um relato de experiência com alunos Iniciantes do Curso de Licenciatura em Computação. In E. Alchieri & P. S. Barreto (Eds.), *Anais do XXXIV Congresso da Sociedade Brasileira de Computação – CSBC 2014: Sistemas Sociais e Eventos de Grandes Massas: Ampliando Desafios da Computação* (pp. 1475-1484). Brasília, DF: Sociedade Brasileira de Computação.

Dias, T. M. B. F. (2014). *A aprendizagem baseada na resolução de problemas com a utilização do "Scratch": um estudo com alunos do 8. º ano de escolaridade no âmbito da PES.* (Dissertação de Mestrado), Universidade Católica Portuguesa, Braga, Portugal. Retrieved from http://hdl.handle.net/10400.14/18081

Dicionário Priberam da Língua Portuguesa. (2013). Disponível em:< http://www.priberam.pt/dlpo/>. *Acesso em, 28*.

Fernandes, J. A. (2014). [Apontamentos da unidade curricular de Metodologia de Investigação em Educação]. Universidade do Minho, Braga, Portugal.

Ferraz, A. P. C. M., & Belhot, R. V. (2010). Taxonomia de Bloom: revisão teórica e apresentação das adequações do instrumento para definição de objetivos instrucionais. *Gestão & Produção, 17*(2), 421-431. doi: 10.1590/S0104-530X2010000200015

Forbellone, A. L. V., & Eberspächer, H. F. (1993). *Lógica de programação: a construção de algoritmos e estruturas de dados* (Vol. 3). São Paulo, Brasil: Makron Books.

Freire, P. (1987). *Pedagogia do oprimido* (17.ª ed. Vol. 3). Rio de Janeiro: Editora Paz e Terra.

Freire, P. (2014). *Educação e mudança*. Rio de Janeiro: Editora Paz e terra.

Furber, S. (2012). Shut down or restart? The way forward for computing in UK schools. *The Royal Society, London*.

Gatti, B. A., & Nunes, M. N. R. (Eds.). (2013). *Formação de professores para o ensino fundamental: estudo de curículos das licenciaturas em pedagogia, lingua portuguesa, matemática e ciências biológicas* (Vol. 29). São Paulo, Brasil: Fundação Carlos Chagas.

Gomes, A. C. G. O. (2013). *Repositório de atividades em Scratch com base nas metas para o 7.º e 8.º ano do ensino básico.* (Dissertação de Mestrado), Instituto Politécnico de Viseu, Viseu. Retrieved from http://repositorio.ipv.pt/handle/10400.19/2024

Gregg, E. A. (2014). *Teaching critical media literacy through videogame creation in scratch programming*. (Ph.D. thesis), Loyola Marymount University. Retrieved from http://digitalcommons.lmu.edu/etd/199

Griebler, G. (2012). Pierre Lévy: as novas tecnologias e a virtualização do mundo humano.

Hoff, M. S. (1996). A matemática na escola nos anos 80-90: Críticas e tendências renovadoras. *Cadernos de Pesquisa, 98*, 72-84.

Hohenwarter, M., & Fuchs, K. (2004). Combination of dynamic geometry, algebra and calculus in the software system GeoGebra. *Computer Algebra Systems and Dynamic Geometry Systems in Mathematics Teaching Conference 2004. Pecs, Hungary*.

Inamdar, P., & Kulkarni, A. (2007). 'Hole-In-The-Wall'Computer Kiosks Foster Mathematics Achievement-A comparative study. *Journal of Educational Technology & Society, 10*(2), 170-179.

International Commission on Education for the Twenty-first Century. (1996). Learning, the Treasure Within: Report to UNESCO of the International Commission on Education for the Twenty-First Century:[summary]: UNESCO.

Jesus, A., & Brito, G. S. (2009). Concepção de ensino-aprendizagem de algoritmos e programação de computadores: A prática docente. *Varia Scientia, 9*(16), 149-158.

Jesus, E. A., & Raabe, A. L. A. (2009). Interpretaçoes da Taxonomia de Bloom no contexto da Programação Introdutória. *Anais do XX Simpósio Brasileiro de Informática na Educaçao*. Florianópolis, SC: SBIE.

Johnson, C. G., & Fuller, U. (2006). Is Bloom's taxonomy appropriate for computer science? *Baltic Sea '06 Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006* (pp. 120-123). New York, NY: ACM.

Jonassen, D. H. (2007). *Computadores, Ferramentas Cognitivas-Desenvolver o pensamento crítico nas escolas*. Porto: Porto Editora.

Kan, S. H. (2002). *Metrics and models in software quality engineering* (2nd ed.). Boston, MA: Addison-Wesley Longman.

Kay, A. (2003). Background on How children learn. *Viewpoints Research Institute*. http://www.vpri.org/pdf/m2003002_how.pdf

Kay, A. (2007). Thoughts About Teaching Science and Mathematics To Young Children. *Viewpoints Research Institute*. http://www.vpri.org/pdf/m2003002_how.pdf

Kishimoto, T. M. (1996). Froebel e a concepção de jogo infantil. *Revista da Faculdade de Educação, 22*(1), 145-167.

Kohl, M. (2015). Lev Vygotsky Coleção Grandes Educadores *Vídeo*. YouTube: ATTA, Midia e Educação.

Krathwohl, D. R. (2002). A revision of Bloom's taxonomy: An overview. *Theory into practice, 41*(4), 212-218.

Laurindo, F. J. B., Shimizu, T., Carvalho, M. M. d., & Rabechini Jr, R. (2001). O papel da tecnologia da informação (TI) na estratégia das organizações. *Gestão & Produção, 8*(2), 160-179.

Lima, M. (2009). Construcionismo de Papert e ensino-aprendizagem de programação de computadores no ensino superior. *UNIVERSIDADE FEDERAL DE SÃO JOÃO DELREI, MINAS GERAIS–BRASIL*.

Lister, R. (2000). On blooming first year programming, and its blooming assessment. *ACSE '00 Proceedings of the Australasian conference on Computing education* (pp. 158-162). New York, NY: ACM.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE), 10*(4), Article No. 16.

Maltempi, M. V., & Valente, J. A. (2000). Melhorando e Diversificando a Aprendizagem via Programação de Computadores. *Proceedings of the ICECE 2000: II International Conference on Engineering and Computer Education*. São Paulo, Brasil: SENAC.

Manzano, J. A. N. G., & Oliveira, J. F. (2000). *Algoritmos: lógica para desenvolvimento de programação de computadores*. São Paulo, Brasil: Érica.

Marques, M. C. P. O. (2013). O ensino da programação no desenvolvimento de jogos através do ambiente Scratch.

Martins, A. R. Q. (2012). *Usando o Scratch para potencializar o pensamento criativo em crianças do Ensino Fundamental.* (Dissertação de Mestrado), Universidade de Passo Fundo, Passo Fundo, Rio Grande do Sul, Brasil. Retrieved from http://www.upf.br/ppgedu/images/stories/defesa-dissertacao-amilton-rodrigo-de-quadros-martins.PDF

Mayer, R. E. (2002). Rote versus Meaningful Learning. *Theory into practice, 41*(4), 226-232.

McCabe, T. J. (1976). A Complexity Measure. *IEEE Transactions on Software Engineering, SE-2*(4), 308 - 320. doi: 10.1109/TSE.1976.233837

Meira, L., & Pinheiro, M. (2012). Inovação na escola. *XI Simpósio Brasileiro de Games e Entretenimento Digital: Anais do SBGAMES*. Brasília, DF: Sociedade Brasileira de Computação.

Mendonça, A. (2010). *Programação Orientada ao Problema: uma Metodologia para Entendimento de Problemas e Especificação no Contexto de Ensino de Programação para Iniciantes.* (Tese de Doutorado), Universidade Federal de Campina Grande, Campina Grande, Paraíba, Brasil. Retrieved from http://docs.computacao.ufcg.edu.br/posgraduacao/teses/2010/Tese_AndreaMendonca.pdf

Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review, 63*(2), 81-97. doi: 10.1037/h0043158

Ministério da Educação, Brasil. (2000). *PCN Parâmetros Curriculares Nacionais (Ensino Médio)*. Brasília: Secretaria de Educação Básica Retrieved from http://portal.mec.gov.br/seb/arquivos/pdf/14_24.pdf.

Ministério da Educação, Brasil. (2002). *PCN+ ensino médio: orientações educacionais complementares aos Parâmetros Curriculares Nacionais: Ciências da Natureza, Matemática e suas Tecnologias*. Brasília: Secretaria de Educação Básica Retrieved from http://portal.mec.gov.br/seb/arquivos/pdf/CienciasNatureza.pdf.

Mitra, S. (2000). *Minimally invasive education for mass computer literacy*. Paper presented at the CRIDALA 2000 conference, 21-25 June, 2000, Hong Kong, China. http:// www.hole-in-the-wall.com/docs/Paper01.pd

Mitra, S., & Dangwal, R. (2010). Limits to self-organising systems of learning—the Kalikuppam experiment. *British Journal of Educational Technology, 41*(5), 672-688.

Morais, M. F. (2012). Criatividade: investimento pessoal e organizacional para o séc. XXI? In L. Faria, M. F. Morais, E. S. Sampaio, J. C. Pinto, A. D. Silva & A. M. D. d. C. Araújo (Eds.), *Actas da VII Conferência de desenvolvimento Vocacional: Carreira, Criatividade e*

Empreendedorismo (pp. 65-82). Braga, Portugal: Associação Portuguesa para o Desenvolvimento da Carreira (APDC).

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Análisis Automático de Proyectos Scratch para Evaluar y Fomentar el Pensamiento Computacional. *Revista de Educación a Distancia, 46*(10). doi: 10.6018/red/46/10

Moysés, L. (1997). *Aplicações de Vygotsky à educação matemática*. Campinas, SP, Brasil: Papirus Editora.

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington, DC: National Research Council.

National Research Council. (2011). *Report of a workshop of pedagogical aspects of computational thinking*. Washington, DC: National Academies Press.

OCDE. (2012). *Relatório Nacional PISA 2012: Resultados brasileiros*. São Paulo, Brasil: Fundação Santillana.

Oliveira, M. K. (1993). *Vygotsky: Aprendizado e desenvolvimento: um processo sócio-histórico*. São Paulo, Brasil: Scipione.

Oliveira, M. L. S., Souza, A. A., Barbosa, A. F., & Barreiros, E. F. S. (2014). Ensino de lógica de programação no ensino fundamental utilizando o Scratch: um relato de experiência. In E. Alchieri & P. S. Barreto (Eds.), *Anais do XXXIV Congresso da Sociedade Brasileira de Computação – CSBC 2014: Sistemas Sociais e Eventos de Grandes Massas: Ampliando Desafios da Computação* (pp. 1525-1534). Brasília, DF: Sociedade Brasileira de Computação.

Onuchic, L. D. L. R., & Allevato, N. S. G. (2011). Pesquisa em Resolução de Problemas: Caminhos, avanços e novas perspectivas. *Bolema-Boletim de Educação Matemática, 25*(41), 73-98.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. New York, NY: Basic Books.

Papert, S. (1997). Educational computing: How are we doing? *T H E Journal, 24*(11), 78-80.

Papert, S. (2002). Hard Fun. Bangor Daily News. Bangor, ME.

Papert, S., & Harel, I. (1991). *Constructionism*. Norwood, NJ: Ablex Publishing.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*(2), 137-168.

Pinto, A. S. (2010). *Scratch na aprendizagem da Matemática no 1. º Ciclo do Ensino Básico: Estudo de caso na resolução de problemas.* (Tese de Mestrado), Universiade do Minho, Braga. Retrieved from http://repositorium.sdum.uminho.pt/bitstream/1822/14538/1/tese.pdf

Pinto, D. G., Costa, M. A., & Marques, M. L. A. (Eds.). (2013). *O Índice de Desenvolvimento Humano Municipal Brasileiro*. Brasília, Brasil: PNUD Brasil.

Postman, N. (2007). *O fim da educação: Redefinindo o valor da Escola* (C. Alcobia, Trans.). Lisboa: Relógio d'Água.

Prata, C. L., & Nascimento, A. C. A. d. A. (2007). *Objetos de aprendizagem: uma proposta de recurso pedagógico*. Brasília: MEC, SEED.

Prensky, M. (2000). Digital natives, digital immigrants part 1. *On the Horizon, 9*(5), 1-6. doi: 10.1108/10748120110424816

Rego, M. C. (2000). *Vygotsky: uma perspectiva histórico cultural da educação*. Petrópolis, RJ, Brasil: Editora Vozes.

Resnick, M. (1998). Technologies for lifelong kindergarten. *Educational Technology Research and Development, 46*(4), 43-55.

Resnick, M. (2002). Rethinking learning in the Digital Age. In G. Kirkman (Ed.), *The Global Information Technology Report: Readiness for the Networked World* (pp. 32-37). Oxford, UK: Oxford University Press.

Resnick, M. (2007). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. *C&C '07: Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition* (pp. 1-6). New York, NY: ACM.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Silverman, B. (2009). Scratch: programming for all. *Communications of the ACM, 52*(11), 60-67.

Ricciardi, L. (n.d.). A Glimpse of the School of Tomorrow? Retrieved from Seymour Papert website: http://www.papert.org/articles/MaineYouthCenterArticle.html

Roseira, N. A. F. (2004). *Educação Matemática e valores: das concepções dos professores à construção da autonomia.* (Dissertação de Mestrado), Universidade do Estado da Bahia - UNEB, Salvador, Brasil.

Sá-Silva, J. R., Almeida, C. D., & Guindani, J. F. (2009). Pesquisa documental: pistas teóricas e metodológicas. *Revista Brasileira de História & Ciências Sociais, 1*(1), 1-15.

Sá, A. L. (2010). Um olhar sobre a abordagem educacional de Reggio Emilia. *Paidéia, 7*(8), 55-80.

Santos, B. F. R. (2014). *Gamification: Uso do scratch no processo de ensino e aprendizagem das Tecnologias da Informação e Comunicação no 8.º ano de escolaridade.* (Tese de Mestrado), Universidade do Minho, Braga. Retrieved from http://repositorium.sdum.uminho.pt/bitstream/1822/38060/1/Bruno%20Filipe%20Ramos%20dos%20Santos.pdf

Santos, M. C. d. (2002). Algumas concepções sobre o ensino-aprendizagem de matemática. *Educação matemática em revista, 12*, 11-15.

Santos, R. M. S. (2013). *Ensino da programação através de programação visual.* (Relatório da Prática de Ensino Supervisionada), Universidade de Lisboa, Lisboa. Retrieved from http://repositorio.ul.pt/bitstream/10451/9137/1/ulfpie044657_tm.pdf

Scaico, P. D., de Lima, A. A., Azevedo, S., da Silva, J. B. B., Raposo, E. H., Alencar, Y., . . . Scaico, A. (2013). Ensino de Programação no Ensino Médio: Uma Abordagem Orientada ao Design com a linguagem Scratch. *Revista Brasileira de Informática na Educação, 21*(02), 92.

Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. *ICER '13 Proceedings of the ninth annual international ACM conference on International computing education research* (pp. 59-66). New York, NY: ACM.

Selby, C. C. (2014). *How can the teaching of programming be used to enhance computational thinking skills?* (Ph.D. thesis), University of Southampton, Southampton, UK.

Senne, E. L. F. (2006). *Primeiro curso de programação em C*. Florianópolis, Brasil: Visual Books.

Shaffer, D. W., & Gee, J. P. (2012). The right kind of GATE: Computer games and the future of assessment. In M. Mayrath, D. Robinson & J. Clarke-Midura (Eds.), *Technology-based assessments for 21st century skills: Theoretical and practical implications from modern research*. Charlotte, NC: Information Age Publications.

Silva Aguiar, G., & Ortigão, M. I. R. (2012). Letramento em Matemática: Um estudo a partir dos dados do PISA 2003. *Boletim de Educação Matemática, 26*(42 A), 1-21.

Silva, M. (2000). *Sala de aula interativa*: Quartet.

Silva, O. M. P., & Panhoca, L. (2007). A contribuição da vulnerabilidade na determinação do índice de desenvolvimento humano: estudando o estado de Santa Catarina. *Ciência & Saúde Coletiva, 12*(5), 1209-1219.

Smole, K. S., & Diniz, M. I. (2001). *Ler, escrever e resolver problemas: Habilidades básicas para aprender Matemática*. Porto Alegre, Brasil: Artmed Editora.

Sousa, R. M., & Lencastre, J. A. (2013). Desenvolvimento do pensamento computacional com recurso ao scratch: Uma experiência com alunos do 8.º ano. In B. D. Silva, L. S. Almeida, A. Barca, M. Peralbo, A. Franco & R. Monginho (Eds.), *Atas do XII Congresso Internacional Galego-Português de Psicopedagogia. Braga: Universidade do Minho* (pp. 6699-6708). Braga: CIEd.

Sousa, R. M., & Lencastre, J. A. (2014). Scratch: Uma opção válida para desenvolver o pensamento computacional e a competência de resolução de problemas. In A. A. A. Carvalho, S. Cruz, C. G. Marques, A. Moura & I. Santos (Eds.), *Atas do 2.º Encontro sobre Jogos e Mobile Learning* (pp. 256-267). Braga: CIEd.

Souza, V. V. S. (2007). *Letramento digital contextualizado: uma experiência na formação continuada de professores.* (Dissertação de Mestrado), Universidade Federal de Uberlândia, Uberlândia, MG, Brasil. Retrieved from http://repositorio.ufu.br/bitstream/123456789/2339/1/LetramentoDigitalContextualizado.pdf

Spivey, G. (2007). A taxonomy for learning, teaching, and assessing Digital Logic Design. *37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports* (pp. F4G-9-F4G-14). Los Alamitos, CA: IEEE.

Thiollent, M. (2011). *Metodologia da pesquisa-ação* (18.ª ed.). São Paulo, Brasil: Cortez Editora.

Tikhomirov, O. K. (1981). The psychological consequences of computerization. In J. V. Wertsch (Ed.), *The concept of activity in Soviet Psychology* (pp. 256-278). Armonk, NY: M. E. Sharpe.

Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., & Verno, A. (2003). A model curriculum for K–12 computer science: Final Report of the ACM K–12 Task Force Curriculum Committee. Computer Science Teachers Association, New York, NY.

Turkle, S. (1997). Seeing through computers. *The American Prospect, 8*(31), 76-82.

Valente, J. A. (1993). Diferentes usos do computador na educação. *Em aberto, 12*(57), 3-16.

Valente, L. (2011). Integração das TIC na educação: o caso do Squeak Etoys.

Valente, L., & Osório, A. J. (2009). What if You Created Your Own Digital Adventurers' Park? *2009 Seventh International Conference on Creating, Connecting and Collaborating through Computing* (pp. 78 - 83). Kyoto, Japan: IEEE.

Vilela, V. V. (2002). Distinções - enriquecendo a vida: Mais distinções, mais opções. from http://www.possibilidades.com.br/percepcao/distincoes.asp

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology, 25*(1), 127-147.

Whalley, J. L., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P. K., & Prasad, C. (2006). An Australasian study of reading and comprehension skills in novice programmers, using the bloom and SOLO taxonomies. *ACE '06 Proceedings of the 8th Australasian Conference on Computing Education* (Vol. 52, pp. 243-252). Darlinghurst, Australia: Australian Computer Society.

Wielewski, G. D. (2008). O movimento da matemática moderna e a formação de grupos de professores de matemática no Brasil. *Actas do XXIV Encontro Nacional de Professores*

de Matemática – ProfMat 2008 (CD-ROM) (pp. 1-10). Lisboa: Associação de Professores de Matemática.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. doi: 10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A, 366*, 3717-3725. doi: 10.1098/rsta.2008.0118

Wing, J. M. (2014, January 10). Computational thinking benefits society [Web log post]. Retrieved from http://socialissues.cs.toronto.edu/index.html%3Fp=279.html

Zorzan, A. S. L. (2012). Ensino-Aprendizagem: Algumas tendências na educação matemática. *Revista de Ciências Humanas, 8*(10), 77-94.

Zulatto, R. B. A. (2007). *A natureza da aprendizagem matemática em um ambiente online de formação continuada de professores.* (Tese de Doutorado), Universidade Estadual Paulista, Rio Claro, SP.